João Pedro Ballerini Bruno

Protocolo USB/IP aplicado em contexto educacional para controle remoto de kits de desenvolvimento com FPGA

São José dos Campos

Fevereiro de 2021

João Pedro Ballerini Bruno

Protocolo USB/IP aplicado em contexto educacional para controle remoto de kits de desenvolvimento com FPGA

Trabalho de graduação apresentado ao Instituto de Ciência e Tecnologia da Universidade Federal de São Paulo, como requisito parcial para obtenção do título de Engenheiro da Computação.

Universidade Federal de São Paulo

Instituto de Ciência e Tecnologia

Bacharelado em Engenharia da Computação

Orientador: Prof. Dr. Tiago de Oliveira

Coorientador: Prof. Dr. André Marcorin de Oliveira

São José dos Campos

Fevereiro de 2021

João Pedro Ballerini Bruno
Protocolo USB/IP aplicado em contexto educacional para controle remoto de kits de desenvolvimento com FPGA/ João Pedro Ballerini Bruno. – São José dos Campos, Fevereiro de 2021-
79 p.

Orientador: Prof. Dr. Tiago de Oliveira
Coorientador: Prof. Dr. André Marcorin de Oliveira

Trabalho de Graduação – Universidade Federal de São Paulo
Instituto de Ciência e Tecnologia
Bacharelado em Engenharia da Computação, Fevereiro de 2021.

1. Aplicação de protocolo *USB/IP* em laboratório remoto. 2. Desenvolvimento de funcionalidade em simulador que o tornou em um laboratório híbrido. 3. Pesquisa de usabilidade da solução. I. Prof. Dr. Tiago de Oliveira. II. Universidade Federal de São Paulo. III. Bacharel em Engenharia da Computação. IV. Protocolo USB/IP aplicado em contexto educacional para controle remoto de kits de desenvolvimento com FPGA

João Pedro Ballerini Bruno

Protocolo USB/IP aplicado em contexto educacional para controle remoto de kits de desenvolvimento com FPGA

Trabalho de graduação apresentado ao Instituto de Ciência e Tecnologia da Universidade Federal de São Paulo, como requisito parcial para obtenção do título de Engenheiro da Computação.

Trabalho aprovado. São José dos Campos, 2 de fevereiro de 2021:

Prof. Dr. Tiago de Oliveira
Orientador

Prof. Dr. André Marcorin de Oliveira
Coorientador

São José dos Campos
Fevereiro de 2021

Este trabalho é dedicado aos professores que, desde sempre, vêm motivando os alunos a criarem coisas extraordinárias.

Agradecimentos

Os agradecimentos são direcionados aos professores orientadores que apoiaram não somente a ideia, mas também em seu desenvolvimento estando sempre próximos as discussões que moldaram o trabalho e também aos colegas que ajudaram durante os testes da aplicação e em questões de *design* e usabilidade.

“Every day do something that will inch you closer to a better tomorrow.“

— Doug Firebaugh

Sumário

RESUMO

Este trabalho foi realizado com o propósito de apresentar uma nova solução para auxiliar no aprendizado de sistemas lógicos em um contexto universitário. Integrado a um simulador de circuitos digitais, a solução foi desenvolvida, trazendo a experiência do laboratório prático para dentro de um ambiente híbrido (simulado e remoto). A interface do aluno com um kit de desenvolvimento com *FPGA* se dá por meio da solução comercial *VirtualHere*, que é capaz de conectar o kit *DE2-115* ao computador do aluno de forma remota e transparente. Esta nova abordagem promete ser menos restritiva e mais responsiva em comparação a solução utilizada atualmente na *Universidade Federal de São Paulo*. Para isto, foi proposta uma nova arquitetura para atingir estes objetivos. Ao término do trabalho o sistema foi disponibilizado para alunos e ex-alunos os quais avaliaram a solução por meio de um questionário de usabilidade *SUS*, obtendo uma pontuação satisfatória.

Palavras-chave: Laboratório Remoto. Laboratório Híbrido. Ensino à Distancia. VirtualHere. USB/IP. FPGA. DE2-115. Arduino. ATmega2560. Raspberry Pi. System Usability Scale. WiRED Panda.

1 Introdução

1.1 Contextualização

Ao avançar dos anos, a revolução digital continua a transformar a vida das pessoas, trazendo diversas mudanças de hábitos para toda a população. Dentre essas transformações, podemos citar a transformação do mercado de trabalho, a inclusão gerada pela facilidade na realização de conferências remotas, as mudanças no ensino superior que contam agora com um público maior e um conteúdo técnico cada vez mais extenso (RASCHKE, 2003). Devido as mudanças nos hábitos, poucas são as tarefas que hoje não possam ser realizadas por meio de um sistema computacional.

Focando-se no escopo educacional, o acesso a informação tem sido tão facilitado que os alunos têm se tornado menos dependentes de uma metodologia e passam a ser cada vez mais os protagonistas de seus aprendizados, de forma que os professores tenham suas funções voltadas ao auxílio e direcionamento do conteúdo. Disciplinas predominantemente teóricas podem então ser compreendidas sem a necessidade de interações presenciais, de forma que o acompanhamento do aprendizado possa ser realizado por meio de trabalhos, questionários e apresentações. No entanto, disciplinas predominantemente práticas tornam-se irrealizáveis sem as devidas ferramentas necessárias. É notável que, embora existam simuladores que possam ser utilizados, estes muitas vezes são limitados e apresentam resultados não observados em experimentos em bancada, o que contribui para um empobrecimento do conhecimento prático. Não obstante, seus usos hão de ser incentivados para complementar as práticas em laboratórios reais (VIDAL; MENEZES, 2015).

De modo a suprir esta limitação, universidades têm optado por desenvolver laboratórios que possam ser manuseados de forma remota, sendo assim possível observar os fenômenos empíricos mesmo sem a presença do aluno (ÁLVARES; FERREIRA, 2003). Embora esta solução seja predominantemente acadêmica, fábricas e companhias também visam fazer uso de soluções remotas, de maneira a automatizar parte do processo e garantir qualidade de produção por meio de um controle remoto supervisionado.

Durante o processo de graduação nos cursos de Engenharia da Computação, Elétrica e Ciência da Computação, avalia-se a vasta presença das unidades curriculares voltadas a compreensão e prática de sistemas e circuitos digitais dada as diretrizes curriculares nacionais (MEC, 2016). No contexto da Universidade Federal de São Paulo, os alunos dos cursos voltados à área da computação fazem uso do laboratório de sistemas embarcados, onde utilizam em grande parte de suas formações um kit de desenvolvimento, com o intuito de simular circuitos de sistemas computacionais projetados pelos alunos no decorrer do

curso. Este kit conta com a presença de um *FPGA (Field Programmable Gate Array)*, um dispositivo lógico programável (BRAGGIO, 2014). Esses dispositivos são utilizados em pelo menos cinco unidades curriculares e devido ao alto custo do equipamento, sua aquisição pelos alunos da universidade é impraticável, restringindo o desenvolvimento das tarefas ao horário das disciplinas. Dada a complexidade dos projetos que tomam proporções grandes com o avançar do curso, esta restrição torna-se um fator limitante ao aprendizado.

Com a implantação de um laboratório remoto no *ICT/UNIFESP* (MAYOZ et al., 2020) o qual permite ao aluno realizar testes de forma remota, estes passaram a sentir-se mais confortáveis com o desenvolvimento de projetos mais robustos, visto que poderiam realizar estas atividades sem a necessidade de fazê-las no período das aulas. Embora tenha elevado abruptamente o aproveitamento dos alunos nas disciplinas, o laboratório remoto foi implementado de forma relativamente simples, onde a interação com a placa é realizada por meio de botões virtuais e os resultados são avaliados pelo usuário através de uma câmera que mostra o kit de desenvolvimento. Todo este processo é realizado sobre o protocolo *HTTP* através de um ambiente *Web*, e como consequência, apresenta latência considerável na manipulação, além de consumir um tráfego alto, incompatível com conexões móveis.

Tendo em mente a metodologia japonesa de melhora ativa *Kaizen* discutida por (BRUNET; NEW, 2003), no presente trabalho será realizado o desenvolvimento de uma nova alternativa para o laboratório remoto existente, onde será implantada uma nova funcionalidade na aplicação *WiRed Panda* desenvolvida por alunos da Universidade Federal de São Paulo (WiRED Panda Team, 2020). A nova solução contará com integração do *software* de prateleira *VirtualHere* com o objetivo de melhorar a usabilidade, assim como incluir novas funcionalidades para tornar a experiência do uso do laboratório menos limitada e mais confortável.

1.2 Objetivos

O objetivo geral do trabalho é implementar novas funcionalidades na aplicação *WiRed Panda* de modo a permitir a realização de conexões remotas com dispositivos como o kit de desenvolvimento *DE2-115*. A nova proposta deverá permitir não somente a realização de testes, mas também leitura e escrita nas memórias internas, realizar depurações e possíveis integrações com demais componentes do kit.

Com um laboratório mais robusto, os alunos serão instigados a desenvolver sistemas ainda mais interessantes, contribuindo assim para o enaltecimento de seus conhecimentos e elevação da qualidade de ensino.

Específicos

- De maneira a expandir as ferramentas dos alunos em operar o kit de desenvolvimento, a nova solução poderá ser incrementada pelo usuário para adicionar novos protocolos e componentes.
- A interface de operação deverá ser amigável e passível de uso contínuo, mesmo que por horas.
- Como forma de integrar kits com defeitos na interface de saída, a nova proposta trará uma nova interface de avaliação dos resultados, sem que haja a necessidade de fazer uso de câmera de monitoramento, que além de depender do bom funcionamento dos kits, não é amigável para usos intensos, visto a qualidade da transmissão e fatores relacionados a uma iluminação não constante.
- A responsividade das operações de entrada e saída deverão satisfazer as necessidades dos projetos, possibilitando até mesmo utilizar componentes de operação síncrona em baixa frequência.
- O projeto deverá ser de baixo custo, e mesmo que exista um custo envolvido, este deverá ser ao menos inferior a aquisição de novos kits de desenvolvimento.
- Não será limitado a apenas um usuário por vez, pois esta solução agora poderá gerenciar múltiplos kits de desenvolvimento simultaneamente.
- Os kits de desenvolvimento poderão ser alternativos, não havendo necessidade de se restringir ao *DE2-115*.

1.3 Organização da monografia

Este trabalho está organizado da seguinte maneira. São apresentados alguns conceitos importantes para o desenvolvimento no Capítulo 2. Uma breve descrição da arquitetura está descrita no Capítulo 3 juntamente com as técnicas e ferramentas utilizadas, separado em seções, neste está abordado também os materiais utilizados e o questionário de avaliação de usabilidade.

A descrição do desenvolvimento do trabalho se dá a partir do Capítulo 4, onde inicialmente é retomado a motivação do trabalho afim de justificar a arquitetura utilizada. Em seguida a arquitetura é apresentada e as partes específicas do sistema são explicadas brevemente, trazendo maior detalhamento no decorrer das seções deste capítulo.

O Capítulo 5 apresenta os resultados obtidos, onde foram feitas duas análises, uma referente a latência de uso do sistema em diferentes cidades e a segunda a partir de uma

pesquisa feita com alunos e *ex-alunos* que fizeram uso da solução, utilizando o questionário apresentado na seção 3.2.

Por fim, no último capítulo, foi apresentada uma breve síntese do trabalho realizado, mas desta vez inserindo-o a uma nova classificação em detrimento de algumas características da solução. Foram comentados os problemas encontrados e as dificuldades na execução do trabalho assim como os próximos passos a serem realizados de modo a melhorar a solução ou torná-la mais acessível.

2 Revisão bibliográfica

2.1 Tipos de laboratórios

2.1.1 Laboratório virtuais

Também conhecido como simuladores, os laboratórios virtuais possuem como propósito imitar os experimentos avaliados na realidade. Geralmente estes são de fácil manuseio e passam uma maior sensação de segurança ao usuário que o manipula, além de apresentar uma visão geral quanto ao funcionamento do experimento (VIDAL; MENEZES, 2015).

Laboratórios virtuais e simuladores atraem inclusive a atenção de renomados físicos, como é o caso de Carl Wieman laureado em 2001 com o Nobel de Física. Em 2002, Wieman fundou o projeto *PhET Interactive Simulations (Physics Education Tecnology)*, com o objetivo de avançar a ciência, literatura matemática e a educação ao redor do mundo através de simulações interativas (BOULDER, (acessado em 22 de fevereiro de 2021)). Os alunos veem os experimentos através de simuladores interativos, assim como os cientistas veem seus experimentos de pesquisa. Cientistas avaliam as pesquisas como formas de explorar conceitos, desafiá-los, corrigi-los, assim como adicioná-los as suas compreensões quanto ao funcionamento do mundo. De forma similar, os alunos exploram as simulações de um modo divertido e através desta, descobrem novas ideias e conceitos sobre a ciência (WIEMAN; ADAMS; PERKINS, 2008).

2.1.2 Laboratórios remotos

Por definição, laboratório remoto é aquele que pode ser utilizado de forma totalmente remota, sendo possível alterar parâmetros de entrada e analisar respostas empíricas. Geralmente os laboratórios remotos, ou *on-line*, são utilizados através de câmeras de modo que possa ser supervisionado e passar uma experiência similar a estar presente em um laboratório prático (LOURENÇO, 2014).

Desde 2001, foi constado que o uso de laboratórios remotos enaltecem a vontade dos alunos de se aprofundarem cada vez mais na disciplina (LINDSAY et al., 2007). A capacidade de realizar os experimentos sem a necessidade de estar presente no laboratório instiga os alunos a realizar as experimentações em momentos extra-classe, assim como possibilita a revisão e a realização de experimentos, mesmo em caso de ausência em aulas presenciais.

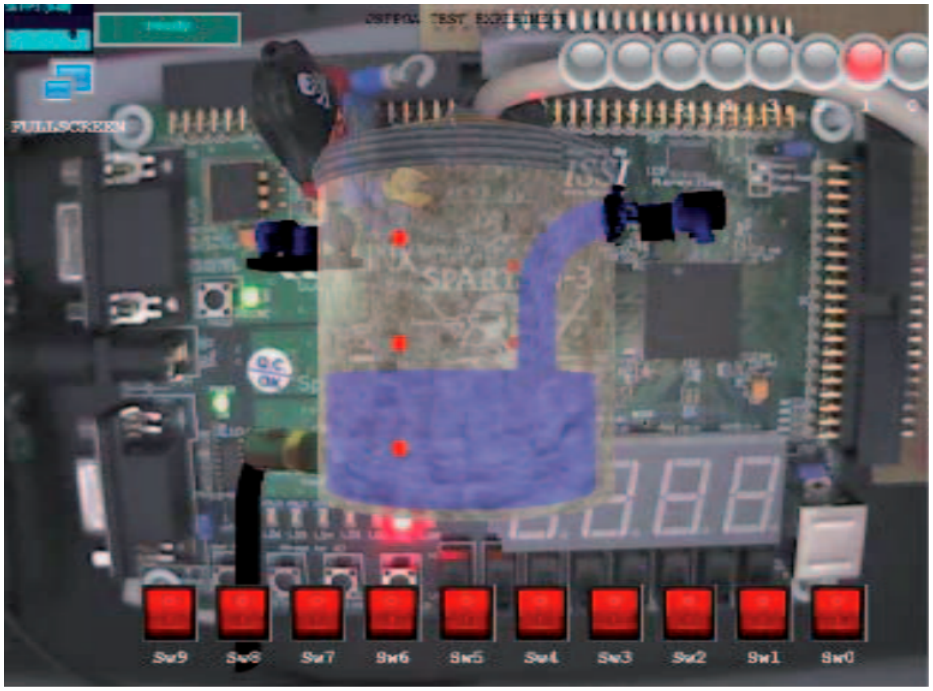
Também se discute quanto a real necessidade de fazer uso de um laboratório remoto,

uma vez que existe também a possibilidade do uso de simuladores ou laboratórios virtuais que prometem ser menos limitados e mais amigáveis para o aprendizado. Embora soluções virtuais contribuam positivamente para o aprendizado, estas muitas vezes são insuficientes para simular o comportamento e as dificuldades encontradas na realização de experimentos na prática, que contribuem no desenvolvimento de uma visão mais crítica da ciência (VIDAL; MENEZES, 2015).

2.1.3 Laboratórios híbridos

Como o próprio termo híbrido já enuncia, os laboratórios híbridos são aqueles que possuem aspectos remotos e virtuais. Um exemplo de laboratório híbrido é o *WebLab-FPGA-Watertank* (Rodriguez-Gil et al., 2014). Neste, o usuário pode programar um kit de desenvolvimento com *FPGA* através do envio do código fonte em linguagem de descrição de *hardware*, após programado, o kit de desenvolvimento remoto passa a controlar um modelo virtual, neste caso o controle de uma caixa d'água. Estes laboratórios, embora pouco presentes, prometem trazer um pouco dos dois mundos colocando a atividade prática nas mãos do aluno e o direcionando para aplicações em diferentes cenários. A interface de operação é apresentada na Figura 1.

Figura 1 – Interface do *WebLab-FPGA-Watertank*



Fonte: Rodriguez-Gil et al. (2014)

2.2 Microcontroladores e microprocessadores

Ambos os componentes são responsáveis por executar atividades de um equipamento, seja gerenciando estados de elementos periféricos (*LEDs*, *Displays*, chaves e botões) ou realizando algum trabalho computacional que requer o processamento de algum algoritmo para atingir a um resultado desejado.

Os microprocessadores geralmente apresentam apenas a lógica necessária para processar instruções. Como estes geralmente não apresentam memória, cristal oscilador ou conversores analógico-digitais, então seu uso é geralmente realizado quando há uma preocupação com desempenho, ou quando o sistema apresenta maior complexidade havendo a necessidade de utilizar configurações específicas. Dispositivos onde há necessidade de incluir diferentes interfaces de comunicação como monitores, telas sensíveis ao toque, memórias principais e secundárias de maior capacidade e desempenho assim como conexões com redes sem-fio ou cabeada, esses fazem uso de microprocessadores e adicionam estas funcionalidades de forma separada, assim havendo a necessidade de uma maior engenharia para utilizá-los devido a primordialidade de um circuito mais robusto para a integração das funcionalidades básicas e das necessárias para o projeto.

Já os microcontroladores são o contrário, estes apresentam não somente a lógica para processar instruções, mas também já possuem de forma integrada todos os demais componentes necessários para seu funcionamento, seja o cristal oscilador (que gera a frequência de operação), ou as diferentes memórias necessárias. O microcontrolador já possui tudo isto integrado em um único chip. Embora seu uso seja mais facilitado, este pode não servir ao propósito do projeto devido sua generalidade, ou devido ao baixo desempenho dada a simplificação da arquitetura e a necessidade de agregar ao mesmo chip diversas outras funções.

Entre os fatores discutidos, deve-se notar que o preço dos componentes variam bastante, sendo os microcontroladores mais baratos que os microprocessadores, isto devido sua simplificação e generalização.

A compreensão das diferenças destes componentes também pode ser avaliada considerando as diversas formas que engenheiros veem esses dispositivos. Suas perspectivas em relação a viabilidade do uso de microcontroladores podem direcionar o desenvolvimento a fazer uso deles devido a não necessidade de realizar *designs* eletrônicos complexos, mantendo assim o baixo custo de desenvolvimento e também de produção. Entretanto, a necessidade de fazer uso de um conjunto de instruções mais complexo para reproduzir algoritmos pode vir a ser uma necessidade para o uso de microprocessadores (WUNDERLICH, 1999).

2.3 *FPGA - (Field Programmable Gate Array)*

O *FPGA (Field Programmable Gate Array)* é um dispositivo lógico programável, que pode ser utilizado para simular circuitos lógicos que antes precisariam ser impressos (BRAGGIO, 2014). Utilizando de uma matriz de transistores que permitem diferentes conexões, o dispositivo tem grande participação em testes de protótipos de circuitos digitais, assim como em dispositivos de alto custo que podem precisar de reprogramação eventualmente.

Comumente encontrado utilizando encapsulamento *QFP (Quad Flat Package)*, *LCC (Leadless Chip Carrier)*, *PLCC (Plastic Leadless Chip Carrier)* ou *OPGA (Organic Pin Gate Array)*, um exemplo destes componentes está presente na Figura 2.

Figura 2 – EP4CE6E22C8N - Cyclone IV

Fonte: RS Components Ltd (acessado em 22 de fevereiro de 2021)

2.3.1 Aplicações do componente

Na atualidade estes componentes estão entrando no mercado de sistemas computacionais profissionais, uma vez que os circuitos por eles sintetizados podem ser atualizados e recombinados de modo a desempenhar diferentes funções específicas. Em termos de *FPGAs*, no evento *WWDC (World-wide developer's conference) - 2019* da *Apple* foi realizado o lançamento de um novo produto da classe de equipamentos profissionais, o *Mac Pro*. Este pode ser obtido com um periférico nomeado *Afterburner* presente na Figura 3 que auxilia no manuseio de vídeos em formato *ProRes* e *ProRes RAW*. Tal periférico utiliza um componente *FPGA* que é capaz de entregar um desempenho muito superior quando comparada a solução por *software*, devido o uso de arquitetura em *Hardware* especializada na execução de um pipeline de atividades específicas.

Figura 3 – Periférico *Afterburner*

Fonte: Apple Inc (acessado em 22 de fevereiro de 2021)

Devido a possibilidade de reprogramação, produtos que realizam interface a nível de circuito lógico entre diversos componentes tendem a fazer uso de *FPGA*, pois caso exista a necessidade de trocar alguma peça por um modelo ligeiramente diferente, o circuito poderá ser reprogramado possibilitando o uso de peças diferentes. Isto abre a possibilidade da empresa fazer uso de peças mais novas mesmo em carros há mais tempo no mercado, poupando-a da necessidade de manter uma fábrica em operação por diversos anos para conseguir dar a devida manutenção a veículos mais antigos (LIU, 2015).

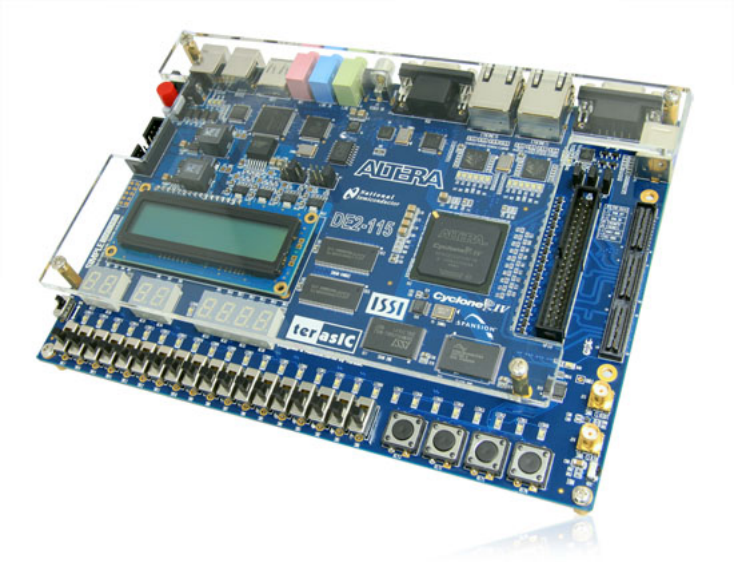
2.3.2 Kit de desenvolvimento com *FPGA*

Devido a capacidade de reprogramação dos componentes *FPGA*, são disponibilizados os kits de desenvolvimento que são soluções integradas destes componentes com diferentes periféricos que podem ser utilizados para testar circuitos lógicos sem a necessidade de realizar a impressão do circuito a cada iteração de testes. Entre estes periféricos estão processadores, interfaces de entrada e saída, dispositivos de memória entre outros.

O kit da *Altera DE2-115* presente na Figura 4 possui o componente *FPGA EP4CE115F29C7 - Cyclone IV* que disponibiliza 114480 elementos lógicos (ALTERA CORPORATION, 2016), sendo este o número de elementos que podem ser recombinados. Entre os dispositivos adicionais presentes neste kit de desenvolvimento estão periféricos de áudio, vídeo, rede *Ethernet*, *displays* de 7-segmentos e *LCD*, *LEDs*, chaves e botões.

Existem diversos outros kits de desenvolvimento para diferentes escopos, porém, no presente trabalho, é utilizado o *DE2-115* devido a disponibilidade na Universidade.

Figura 4 – Kit de desenvolvimento *DE2-115*



Fonte: Terasic Technologies Inc (2010)

2.3.3 Laboratórios de FPGA

No curso de Engenharia Elétrica e Engenharia da Computação, o uso de kits de desenvolvimento com *FPGA* é bastante frequente e ocorre usualmente em laboratórios de sistemas embarcados e de sistemas computacionais. Em (BERALDO; OLIVEIRA; STRINGINI, 2020), foi realizada uma revisão sistemática da literatura onde o enfoque foi colocado em laboratórios remotos de *FPGA* voltados ao ensino. Neste concluiu-se que embora estejam mais presentes em países estrangeiros, o resultado comum obtido por aqueles que adotam seu uso é a satisfação dos alunos.

Alguns pontos importantes avaliados pelo trabalho de revisão bibliográfica foram:

- O motivo de implantação é bastante comum entre as publicações avaliadas tendo como principais objetivos: questões financeiras; prover maior acessibilidade e flexibilidade; motivar os alunos; oferecer formas de acesso à distância ao laboratório.
- Em alguns deles, a interação é feita utilizando uma câmera, em outros, algum software que mostra os resultados obtidos.
- Sistemas de correção automática de atividades já foram implementados.
- A aceitação dos alunos tal para com o uso dos laboratórios é expressiva e estes zelam pela manutenção destas soluções.

3 Metodologia

As metodologias a serem utilizadas irão depender da etapa de desenvolvimento e das tarefas a serem administradas. Dentre as utilizadas, pode-se mencionar a metodologia *Scrum* para gerenciar as atividades voltadas ao desenvolvimento que requerem foco e a metodologia *Kanban*, para gerenciar atividades que são executadas de modo paralelo ao desenvolvimento principal, como a revisão de pontos destacados pelos orientadores, definição de novas atividades e redefinição de escopo (MAHNIC, 2014).

A solução final consiste no desenvolvimento da aplicação e sua disponibilização. Para isto, o desenvolvimento foi dividido em duas etapas: a primeira apresentou um subconjunto das funcionalidades, e serviu para prover os resultados de performance iniciais; a segunda etapa foi discretizada pelo refinamento da solução, trazendo novas funcionalidades e uma interface gráfica mais amigável. Com a ajuda de colegas e docentes, testes foram realizados no decorrer de todo o desenvolvimento e impactaram diretamente nas tomadas de decisão de implementações de modo a melhorar a experiência do usuário.

Além do *WiRedPanda*, que recebeu estas novas funcionalidades, foi necessária a implementação de uma aplicação servidor responsável por gerenciar a autenticação dos usuários, assim como realizar o repasse dos estados dos registradores de entrada e saída entre a plataforma e o microcontrolador *Arduino*. Este serviço irá possuir implementações específicas da organização onde será utilizado, podendo então ser adaptado conforme a necessidade. Seu desenvolvimento deverá ser realizado pensando na agilidade da comunicação e na restrição de desempenho do sistema embarcado *Raspberry PI 4*® em que esta será executada. Esta aplicação também poderá conter integrações com o servidor da aplicação de prateleira *VirtualHere*® que será utilizada para realizar a comunicação *USB/IP*.

A implementação de um novo protocolo será necessária para trabalhar com a conexão persistente *TCP/IP*. Este fará uma comunicação ativa e de forma bidirecional para realizar as atualizações dos estados dos registradores de entrada e saída, e também deverá oferecer suporte para realizar a configuração dos pinos, visto que as placas utilizam pinos *GPIO* e poderão atuar como entrada ou saída de dados.

O protocolo *USB/IP* foi desenvolvido com a finalidade de tornar-se uma alternativa menos limitada e mais veloz de realizar compartilhamento de dispositivos através de uma rede (HIROFUCHI et al., 2005), e por padrão está implementado nos *kernels* atuais da maioria dos sistemas *Linux* como visto em (HIROFUCHI, 2011 (acessado em 8 de outubro de 2020)). No entanto, seu funcionamento é limitado por ser dependente a plataforma e, embora seja possível torná-lo compatível com as demais plataformas, devido a necessidade de fazer alterações a nível de *drivers*, existe um árduo processo a ser seguido para tornar

esta solução plausível. Uma solução alternativa foi desenvolvida pela empresa ©*VirtualHere Pty*. Esta, além de oferecer suporte a diferentes dispositivos com otimizações específicas, oferece aplicações com interface de controle e *APIs* para manipulá-la.

Para gerenciar as entradas e saídas, será necessário também, para cada kit de desenvolvimento com *FPGA* conectado, um microcontrolador que será programado para atuar no controle de entrada e saída. Este receberá instruções do servidor por meio de comunicação *USART* também sobre o protocolo *USB*. Seu uso está relacionado a necessidade de uma quantidade muito grande de entradas e saídas a serem controladas para cada kit de desenvolvimento, devido a limitação da quantidade de pinos *GPIO* (*General Purpose Input & Output*) presente no sistema computacional embarcado, assim como a necessidade de um controle mais robusto sobre as entradas e saídas.

Em suma, a arquitetura da solução é composta de quatro partes principais, a aplicação *WiredPanda* instalada na máquina do usuário, o serviço *VirtualHere* que terá suas customizações e integrações com o sistema, assim como o serviço responsável pela autenticação dos usuários e comunicação entre a aplicação cliente e o controlador programável que efetivamente conectará o kit de desenvolvimento ao serviço. A arquitetura da solução proposta está presente na Figura 5. Esta será melhor discutida na seção de desenvolvimento.

Estratégia de desenvolvimento

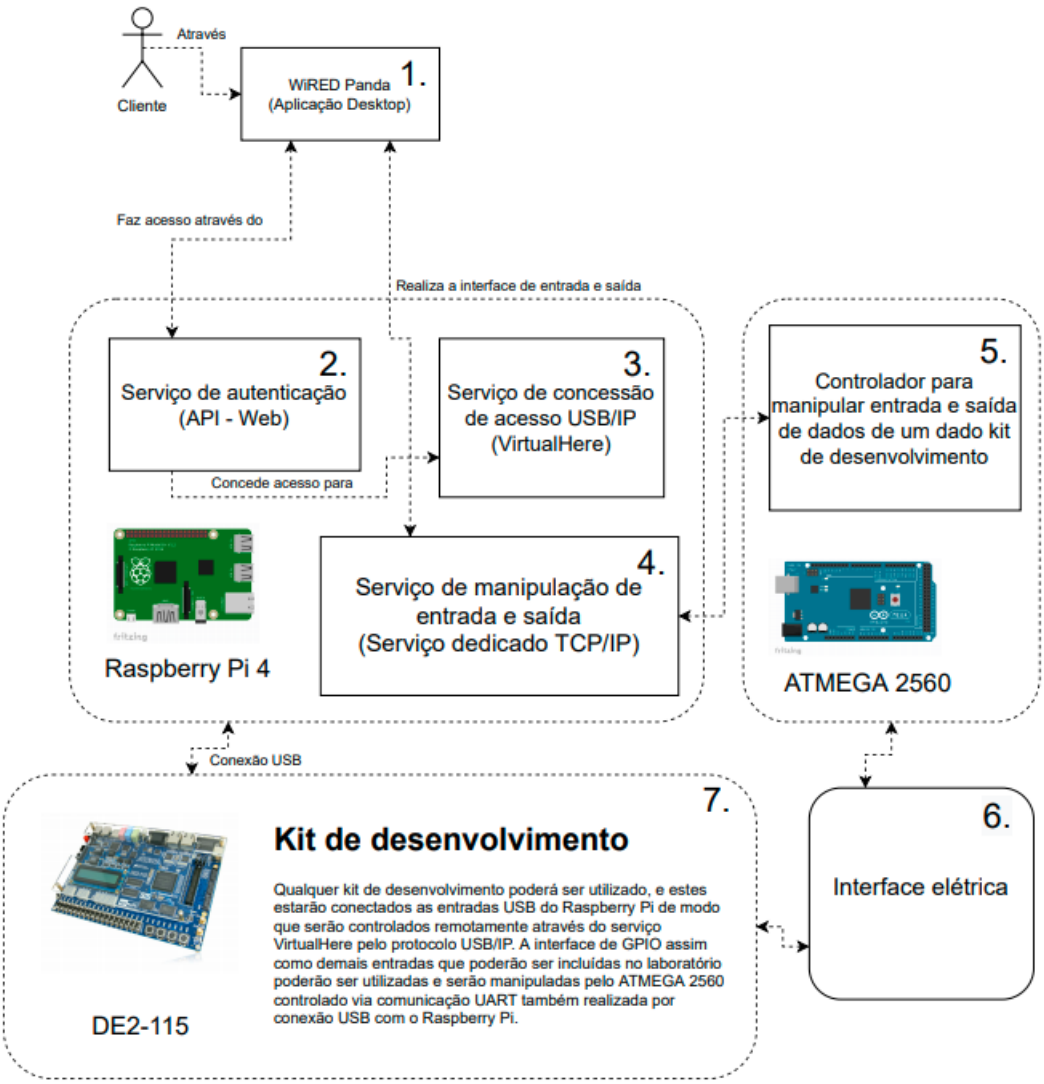
Algumas técnicas foram aplicadas no desenvolvimento do trabalho para se obter agilidade no desenvolvimento e evitar a ocorrência de imprevistos devido a incompatibilidades entre diferentes plataformas (RAJ; TOLETY, 2012).

Embora a *framework* de desenvolvimento *Qt* seja multi-plataforma, ainda assim poderá ocorrer problemas de compatibilidade entre os diferentes sistemas. Para contornar eventuais problemas, o desenvolvimento foi realizado parte em ambiente *Windows* e no *MacOS*.

A fim de garantir o controle de versão, foi utilizado a ferramenta *open-source Git*. Esta possibilitou não somente o armazenamento, mas também o desenvolvimento simultâneo da ferramenta. Sendo assim, durante o desenvolvimento do laboratório remoto, outras funcionalidades foram implementadas na ferramenta, e assim, o *Git* possibilitou a união das diferentes versões e funcionalidades.

Outra técnica utilizada foi realizar a implementação das funcionalidades de modo concorrente, não necessariamente terminando os pequenos entregáveis, mas implementando-os aos poucos. Em ambientes com múltiplos desenvolvedores, esta técnica pode ser impraticável, contudo, quando aplicada a apenas um desenvolvedor, garante a possibilidade de enxergar melhores relacionamentos entre as implementações *on the go* durante o desen-

Figura 5 – Arquitetura da solução proposta



Fonte: Desenhada utilizando ferramenta *DrawIO* com imagens obtidas da aplicação *Fritzing*® e documentação oficial da *Altera*

volvimento. Sendo possível prever problemas de incompatibilidades ou de acoplamento desnecessário, ocorridos devido ao planejamento de tarefas de forma equivocada, podendo corrigi-los antes que o retrabalho necessário seja grande a ponto de haver necessidade de refazer completamente dada atividade.

3.1 Materiais e métodos

Esta seção trará os materiais utilizados, assim como os *softwares* e tecnologias.

3.1.1 *Frameworks* e *softwares* utilizados

Para o desenvolvimento, foi necessário avaliar a tecnologia a se utilizar, que devido ao objetivo de obter desempenho, o desenvolvimento foi realizado sobre uma aplicação *desktop* já existente o qual foi criada fazendo uso de *frameworks* - um conjunto de funcionalidades já implementadas, assim otimizando o tempo e direcionando a apenas o desenvolvimento das funcionalidades específicas.

3.1.1.1 Aplicações *Desktop*

As aplicações *desktop* são aquelas que devem ser instaladas na máquina do usuário para conseguirem operar. Dentre os motivos para se fazer uma aplicação *desktop* estão:

- Necessidade de desempenho;
- Integração com sistemas e serviços do sistema operacional não acessíveis a partir de aplicações *Web*;
- Sistemas que exigem análise de dados em tempo real, e persistência de conexão;
- Soluções que fazem integrações com outras soluções *desktop*;

Como a abordagem será desenvolvida integrada a aplicação *WiredPanda* e também fará uso do *software* comercial *VirtualHere* - uma solução *desktop* para realizar a conexão de dispositivos periféricos *USB* de forma remota - poderemos contar com vantagens como maior desempenho e maior responsividade dada a persistência das conexões, garantindo assim uma melhor experiência para o usuário.

3.1.1.2 Qt

Uma plataforma de desenvolvimento capaz de criar aplicações multi-plataformas, e de forma rápida. Contém uma vasta implementação de protocolos e funcionalidades que permite fácil integração com sistemas de rede, *API* gráfica e elementos gráficos do sistema operacional. Esta *framework*, por ser *OpenSource*, é extremamente confiável e deve ter seu suporte continuado por inúmeros anos.

Para fazer uso desta plataforma existem duas opções de licenças:

- Comercial, onde é necessário fazer a compra dos direitos de publicar o *software Qt* para fins comerciais, onde há a preocupação de tornar o código fonte da aplicação fechado.
- Livre, onde o *software Qt* poderá ser usado de forma irrestrita, desde que tenha seu código fonte disponível e licenciado pela *LGPLv3* (FREE SOFTWARE FOUNDATION, 2007b) ou *GPLv3* (FREE SOFTWARE FOUNDATION, 2007a).

3.1.1.3 WiRedPanda

Um *software* livre licenciado sob a licença *GPL-3.0* (FREE SOFTWARE FOUNDATION, 2007a) que foi desenvolvido na Universidade Federal de São Paulo, para ser um simulador de circuitos de lógica digital intuitivo e fácil de operar, utilizado em ambientes de aprendizados.

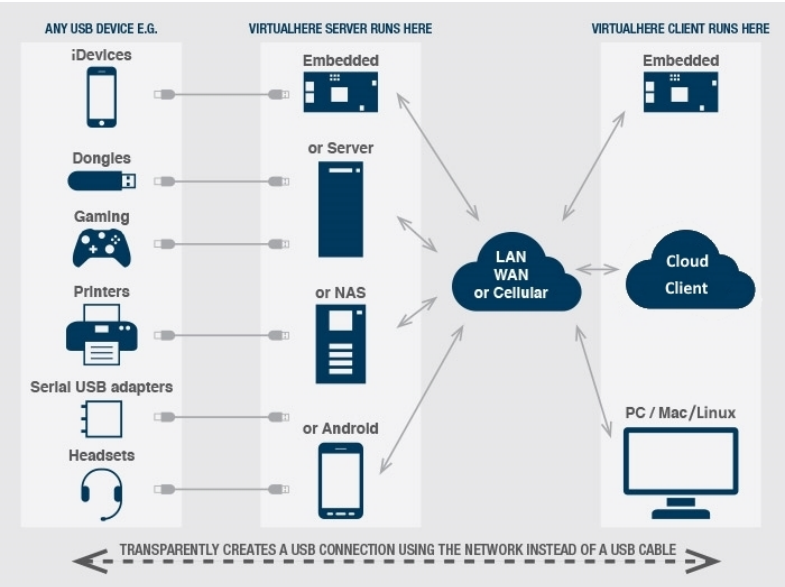
Está licenciada sob a licença livre *GPLv3* (FREE SOFTWARE FOUNDATION, 2007a) e o código fonte está disponível no *GitHub*, plataforma online de versionamento e consulta de códigos.

3.1.1.4 *VirtualHere*

Com o propósito de empregar dispositivos periféricos *USB* de forma remota, a aplicação comercial *VirtualHere* oferece um meio de realizar a conexão dos dispositivos através de uma rede, sendo ideal para evitar que estes precisem estar conectados nos computadores em que são utilizados.

De forma transparente ao usuário, o *VirtualHere* oferece *drivers* disponíveis para os sistemas *Windows*, *Linux* e *MacOS* que virtualmente conectam dispositivos no computador do cliente sem que exista a necessidade destes serem feitos para operar de forma remota. Como o processo é feito a nível de protocolo, este é transparente aos *drivers* dos próprios dispositivos. Caso a velocidade de conexão não seja um pré-requisito, qualquer periférico *USB* que funcionar nos sistemas operacionais poderão ser utilizados de forma remota, mesmo estando conectado em outro computador.

Figura 6 – Diagrama de funcionamento da solução



Fonte: VirtualHere (2021)

Outros trabalhos que fizeram uso desta solução relataram apresentar um bom desempenho, mesmo quando utilizados em redes *internet*. Um exemplo de aplicação do *VirtualHere* no Brasil foi na Universidade Federal de Uberlândia, onde foi criado um laboratório de baixo custo utilizando computadores "magros" (*thin-clients*) que atuam apenas como terminais controlados por um servidor de processamento central (SILVA et al., 2019). A solução possibilitou a integração dos controladores do laboratório nos terminais conforme o uso.

Em (PINTILIE et al., 2018) foi redigido um trabalho que utilizou esta solução para conectar dispositivos periféricos através da rede elétrica. Para isto, foram utilizados adaptadores para criar uma rede *ethernet* através da geração de ruídos de alta frequência na instalação elétrica de modo que possibilitou o desenvolvimento de um *socket* de tomada multi-função.

3.1.2 Materiais utilizados

Os materiais utilizados para a construção deste laboratório foram em parte comprados em território nacional e em parte comprados no exterior. Em uma pesquisa de avaliação de custos, foi possível perceber uma variação muito grande nos preços dos equipamentos, sendo que alguns deles variaram até R$ 800,00 dependendo do local de compra, como é o caso do *Raspberry Pi*.

De forma a manter este documento o mais atualizado possível, os preços dos componentes serão tratados em dólares, considerando compras feitas nos Estados Unidos na rede de compras *Amazon*. Em território nacional, o preço dos componentes podem variar muito caso a compra seja feita em lojas físicas, em distribuidoras ou no exterior, contudo, em importações é possível obter um preço próximo ao avaliado nas compras nos Estados Unidos.

Tabela 1 – Cotação dos componentes necessários

Produto	Loja	Preço (U$)	Recomendado	Mínimo
Raspberry Pi 4 4GB	Amazon (EUA)	61,88	X	
Raspberry Pi 4 2GB	Amazon (EUA)	52,99		X
RPI4 - Case, Heatsinks and Fan	Amazon (EUA)	10,99	X	
RPI4 - Power Supply	Amazon (EUA)	9,99	X	X
microSDHC - 32GB	Amazon (EUA)	7,49	X	X
Arduino Board ATmega 2560	Amazon (EUA)	36,28	X	
ATmega 2560 - Generic	Amazon (EUA)	17,99		X
ELEGOO 830pts Breadboard (3pcs)	Amazon (EUA)	9,99	X	
ELEGOO Jumper Wires (120pcs)	Amazon (EUA)	7,65	X	
Arduino Relay Switch 5V (2pcs)	Amazon (EUA)	5,79	X	
DEV KIT DE2-115 CYCLONE IV	Mouser Electronics	595,00		
DEV KIT DE0-NANO CYCLONE IV	Amazon (EUA)	121,71		
DEV KIT DE0-NANO CYCLONE IV	TerasIC	79,00		X
DEV KIT DE0-NANO-SoC CYCLONE V	TerasIC	99,00	X	
VIRTUALHERE LICENSE	VirtualHere Pty	49,00	X	X
			TOTAL	
			298,06 U$	216,46 U$

Fonte: O Autor

Na Tabela 1 foram inseridos todos os produtos que podem ser utilizados. Contudo há mais de uma opção de um mesmo produto, como é o exemplo do *Raspberry Pi*. O custo foi calculado levando em consideração a compra recomendada dos componentes e a compra mínima - é importante ressaltar que itens como fios *jumpers*, *protoboard* e relês não estão incluídos na aquisição *mínima*, devido o baixo custo destes itens e o fato de estarem disponíveis em praticamente todas as lojas de produtos eletrônicos.

Diversos kits de desenvolvimento com *FPGA* foram inseridos na Tabela 1, visto que seu uso pode variar de acordo com o requisito do projeto. A instalação realizada neste trabalho fez uso do kit *DE2-115*, pois este já estava disponível na Universidade. Contudo não é interessante adquiri-los para este propósito, pois este possui muito mais funcionalidades do que será possível de utilizar. Deste modo foram incluídas outras opções de compra que estão mais alinhadas com o intuito do projeto. Neste caso o kit *DE0-Nano* da *Altera* é mais indicado para este projeto, pois possui uma maior quantidade de pinos acessíveis que poderão ser mapeados para o uso remoto, entretanto, existem duas versões deste kit. A versão mais nova possui um componente *FPGA Cyclone V* (mais recente) e um processador *Cortex-A9* de arquitetura *ARM* integrado. Embora esta versão seja um pouco mais cara, ela irá ampliar as possibilidades de uso do equipamento.

Foi utilizado no trabalho um *ATmega 2560* genérico. Devido a isso, alguns problemas de compatibilidade foram encontrados sendo necessário utilizar *drivers* diferenciados para funcionar. Para prevenção de eventuais incompatibilidades é sugerido o italiano.

Em relação a licença para uso do *VirtualHere*, embora tenha o preço equivalente a

um *Raspberry PI*, seu uso é perpétuo, ou seja, não há a necessidade de renová-la. Entretanto, vale salientar que no caso do servidor deixar de funcionar e for necessário mudanças no *hardware*, estas poderão invalidar a licença.

Quanto ao *Raspberry PI*, não há motivos para selecionar a versão de *4GB* ao invés da versão de *2GB*, isto porque a solução usa menos de *768MB* de memória principal. Contudo, vale lembrar que ao fazer a aquisição da licença *VirtualHere*, será necessário associar a licença a um dispositivo e, como a licença é intransferível, é interessante optar por uma versão superior para que não tenha que fazer uma nova aquisição em caso de necessidade. Por este mesmo motivo, não é recomendado fazer uso de um *Raspberry PI 3B* ou anterior, embora, ao menos para o *VirtualHere* um *Raspberry PI 3B* seja suficiente.

3.2 Questionário de avaliação de usabilidade

De modo a avaliarmos como tem sido a aceitação dos alunos com a nova solução do laboratório remoto, será realizado um questionário com os alunos após suas primeiras interações com o sistema. Esta avaliação será baseada no *SUS* (*System Usability Scale*). Este é um padrão de questionário para avaliar a usabilidade de algum sistema ou equipamento, foi inicialmente apresentado em 1996 por John Brooke (BROOKE, 1996) e sua publicação original possui 10381 citações, segundo a plataforma *Scholar Google* (avaliado no dia 11/02/2021). Este padrão de questionário foi utilizado em 2009 por aproximadamente *40%* das indústrias que realizavam avaliações de usabilidade segundo Sauro e Lewis (SAURO; LEWIS, 2009).

Os pontos os quais são explorados por estas questões avaliam os seguintes critérios:

- Capacidade de aprendizagem e o quão fácil é utilizá-lo pela primeira vez;
- Rapidez com que os resultados esperados são entregues, seja em facilidade de realizar as tarefas, ou em responsividade;
- Intuitividade, sendo assim fácil de memorizar as instruções necessárias para operá-lo;
- Ausência de comportamentos inesperados, seja devido a erros ou relacionado a baixa *QOS* (*Quality of Service*);
- Satisfação gerada pelo uso do sistema, em termos de *design* ou quanto dinamicidade das operações;

Alguns destes critérios foram descritos por Jacob Nielsen (NIELSEN, 2003) e compõem os fatores que levam o sistema a ter baixo ou alto fator de usabilidade, sendo explorado pelo *SUS* (*System Usability Scale*) as questões a serem levantas para considerar estes fatores.

Além de prover perguntas formuladas para avaliar os diversos pontos relevantes da usabilidade, o *SUS* também é capaz de gerar uma pontuação final com base nas respostas, pontuação esta avaliada no intervalo de 0 a 100 que deverá classificar o sistema analisado. Em um trabalho realizado em 2009 (BANGOR; KORTUM; MILLER, 2009), foram relacionadas as pontuações *SUS* com adjetivos que ajudam a melhor compreender o significado destas pontuações, onde foi possível extrair a Tabela 2.

Tabela 2 – Relação da pontuação *SUS* com seus respectivos significados em adjetivos

Adjective	Count	Mean SUS Score	Standard Deviation
Worst Imaginable	4	12.5	13.1
Awful	22	20.3	11.3
Poor	72	35.7	12.6
OK	211	50.9	13.8
Good	345	71.4	11.6
Excellent	289	85.5	10.4
Best Imaginable	16	90.9	13.4

Fonte: (BANGOR; KORTUM; MILLER, 2009)

Outras seis perguntas foram adicionadas ao questionário para fazer uma análise quanto ao contexto em que o aluno fez uso do sistema, sua prospecção quanto ao momento em que este laboratório se tornou disponível e se este foi capaz de utilizá-lo, visto a não compatibilidade da solução com conexões com alta latência ou instáveis.

1. Fui capaz de utilizar o laboratório remoto;
2. Acredito que gostaria de utilizar este sistema frequentemente*;
3. Achei o sistema desnecessariamente complexo*;
4. Achei o sistema fácil de usar*;
5. Acredito que precisaria do apoio de um suporte técnico para fazer uso do sistema*;
6. Achei que as funções presentes neste sistema foram bem integradas*;
7. Achei que houve muitas inconsistências neste sistema*;
8. Imagino que a maioria das pessoas aprenderiam a usar esse sistema rapidamente*;
9. Achei o sistema muito pesado para utilizá-lo*;
10. Me senti bastante confiante usando esse sistema*;
11. Houve a necessidade de aprender uma série de coisas antes que eu pudesse continuar a utilizar esse sistema*;

12. Tive contato com a ferramenta *WiredPanda* anteriormente;
13. Acredito que o momento de disponibilização desta ferramenta é oportuno;
14. Acredito que esta ferramenta torna as possibilidades de desenvolvimento de circuitos menos limitadas, contribuindo positivamente para o desenvolvimento das atividades;
15. Qual foi a tecnologia de rede utilizada para conectar-se a internet durante o uso da ferramenta;
16. Qual dos laboratórios você se encontra em sua trajetória acadêmica;

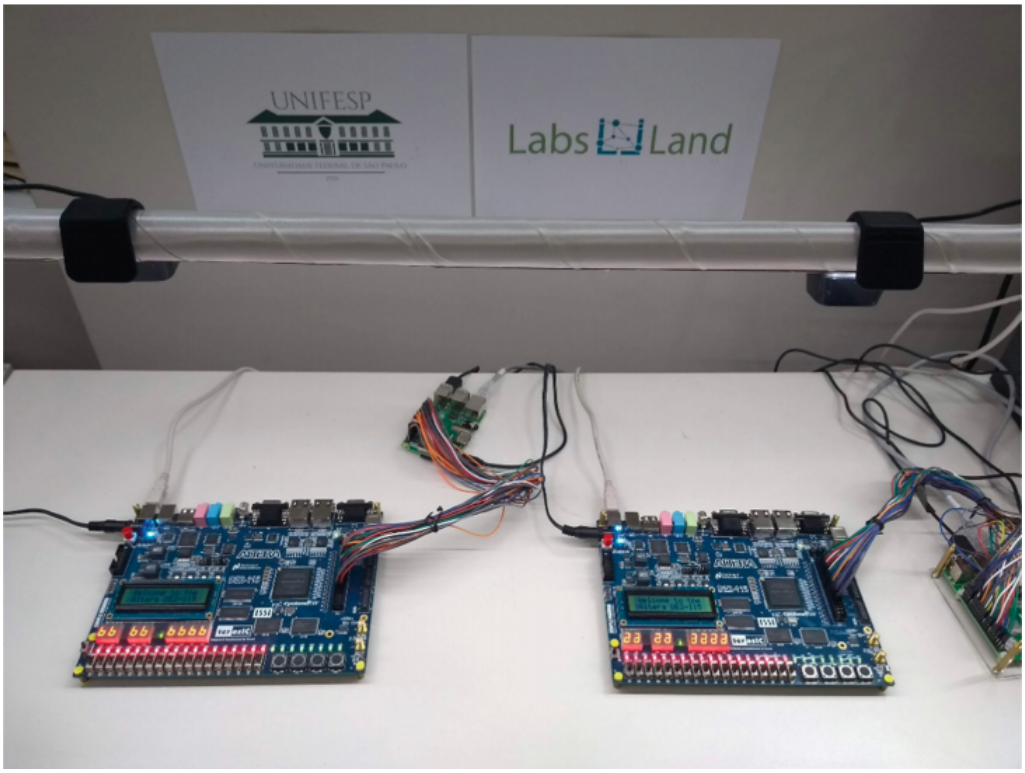
As questões indicadas com um asterisco são as questões do *SUS* e serão avaliadas, como supostas, através da escala *Likert* a ser assinalada pelo respondente. Estas questões serão somente direcionadas ao aluno que responder positivamente a primeira questão. A questão 12 será respondida de forma lógica (*Booleana*) e poderá influenciar na pontuação do *SUS*. Já as questões 13 e 14, deverão ser respondidas pela escala *Likert* e serão utilizadas para traçar paralelos ao contexto. Nas duas últimas questões, serão dadas diferentes opções de escolha que serão importantes para avaliar o impacto da qualidade da conexão do aluno na pontuação de usabilidade e também avaliar o contexto dos alunos que estão fazendo uso desta solução.

4 Desenvolvimento

É importante no início deste capítulo contextualizar a motivação que levou a adoção desta arquitetura. Como descrito na seção 1.1, em um trabalho previamente realizado nesta mesma Universidade (MAYOZ et al., 2020), um projeto similar de laboratório remoto para o uso destas placas *DE2-115* foi proposto e implantado não somente na presente Universidade, mas também na *UPNA* (Universidade Pública de Navarra - Espanha). Ao avaliar a solução previamente apresentada, foi avaliada a possibilidade de trazer melhorias a ideia do laboratório, que pudessem levar a experiência a um novo nível. A filosofia *Kaizen* foi citada anteriormente uma vez que este projeto originou da ideia de tentar melhorar a experiência do aluno.

O projeto anterior fazia uso de uma câmera como método de apresentação de resultados conforme mostrado na Figura 7, que embora dê a experiencia ao aluno de colocá-lo no laboratório, apresentava algumas desvantagens.

Figura 7 – Arquitetura da solução proposta



Fonte: Laboratório remoto da UNIFESP (MAYOZ et al., 2020)

Devido a necessidade de decodificação e *streaming* de vídeo, a solução é pesada em termos de processamento e trabalha com um alto tempo de resposta tornando seu uso um pouco desconfortável. Além disso, a conexão foi realizada utilizando protocolo *HTTP*

na camada de aplicação, que também contribuía negativamente para a a responsividade do sistema. Embora tenha funcionado muito bem e tenha contribuído bastante para o aprendizado dos alunos, estes pontos arquiteturais tornam-se oportunidade de melhoria.

Dado o devido contexto e motivação, é possível compreender que esta nova arquitetura foi proposta para reduzir a latência e enaltecer a responsividade do sistema, assim como aumentar as possibilidades de projeto.

4.1 Visão geral

Devido a complexidade e distribuição de responsabilidades entre diversas aplicações e sistemas computacionais, é importante que seja possível compreender exatamente a função de cada um dos sistemas, para então ser possível tratar detalhes a respeito da implementação em cada caso.

Na Figura 5 presente na seção 3, foram enumerados cada um dos blocos do sistema, de modo a facilitar a explicação de suas relações.

A aplicação *desktop - WiRED Panda* é o primeiro bloco a ser avaliado, pois é a aplicação cliente, de onde todas as requisições partem. Esta aplicação faz contato com dois diferentes serviços hospedados no *Raspberry PI*, o serviço de autenticação (bloco 2) que é uma aplicação *Web* com arquitetura *REST* (*Representational State Transfer*) o qual trabalha com requisições realizadas e respondidas por meio de formulários em formato *JSON*; e o serviço de manipulação de entrada e saída (bloco 4), responsável por atividades que exijam baixa latência para prover a melhor qualidade de serviço (*QOS*). O cliente em um primeiro momento comunica-se somente com o serviço *Web* que irá autenticar o usuário e gerar sua sessão de acesso. Após gerada a sessão, o cliente estabelece uma conexão com o outro serviço, desta vez persistente.

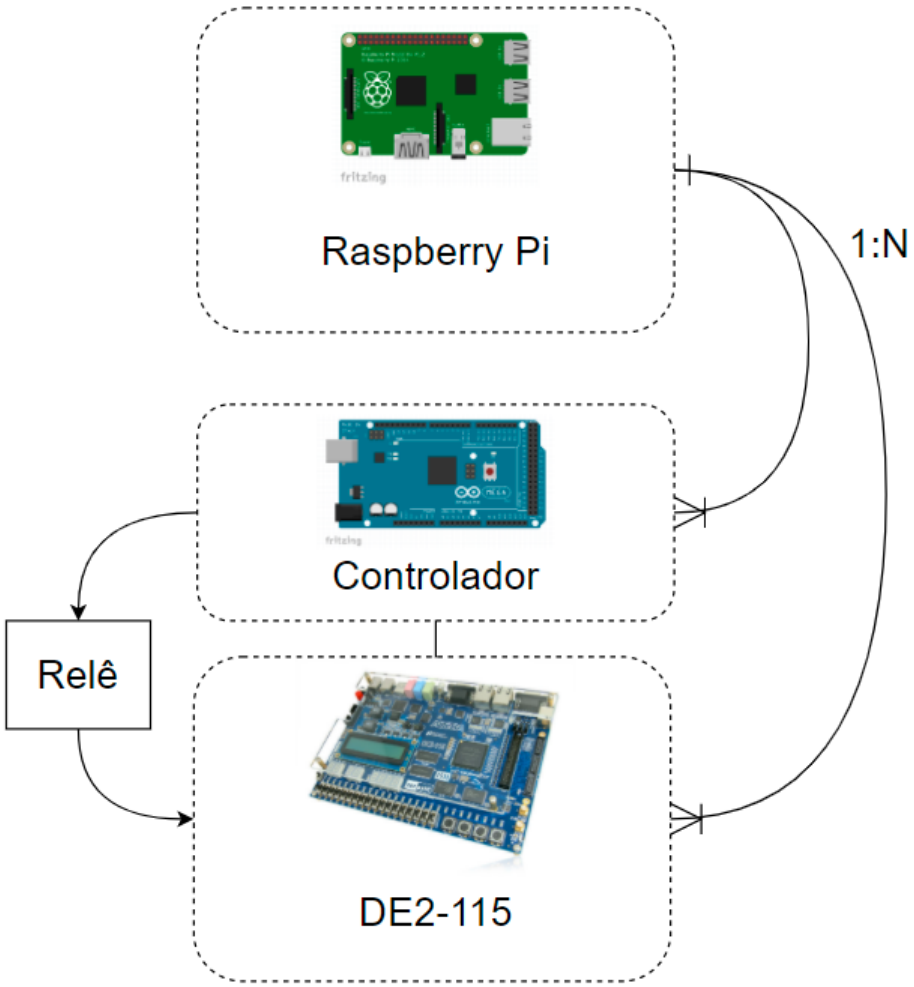
Após gerar a sessão do usuário e concedê-lo um dispositivo, o serviço de autenticação (bloco 2) gera também um *token* para acesso ao dispositivo via cliente *VirtualHere*. A aplicação cliente do *VirtualHere* está presente junto com a aplicação *WiRED Panda* existindo uma integração para abri-la no momento em que a janela de configurações torna-se disponível. Existe também um outro serviço operando no *Raspberry PI*, que é o serviço servidor do *VirtualHere* (bloco 3), responsável por redirecionar o tráfego dos dispositivos *USB* via *USB/IP* para os clientes. Neste serviço, existe a possibilidade de realizar integrações utilizando *scripts Shell*, o qual foram escritos de forma a verificarem o *token* inserido pelo usuário. Esta verificação é feita através de requisições *HTTP* locais direcionadas ao serviço *Web* (bloco 2).

Com o usuário já autenticado, e com uma sessão ativa com um dispositivo, o serviço de manipulação (bloco 4) passa a se comunicar com o respectivo controlador daquele

dispositivo (bloco 5). Através desta comunicação, são passadas as instruções referentes a pinagem utilizada pelo usuário, assim como as instruções referentes a alteração dos estados dos pinos de entrada e saída. Este controlador por sua vez controla os pinos do kit de desenvolvimento *DE2-115* (bloco 7), e para isto, estas ligações passam por uma interface elétrica (bloco 6), que possibilita uma conexão segura entre o micro-controlador e o kit de desenvolvimento.

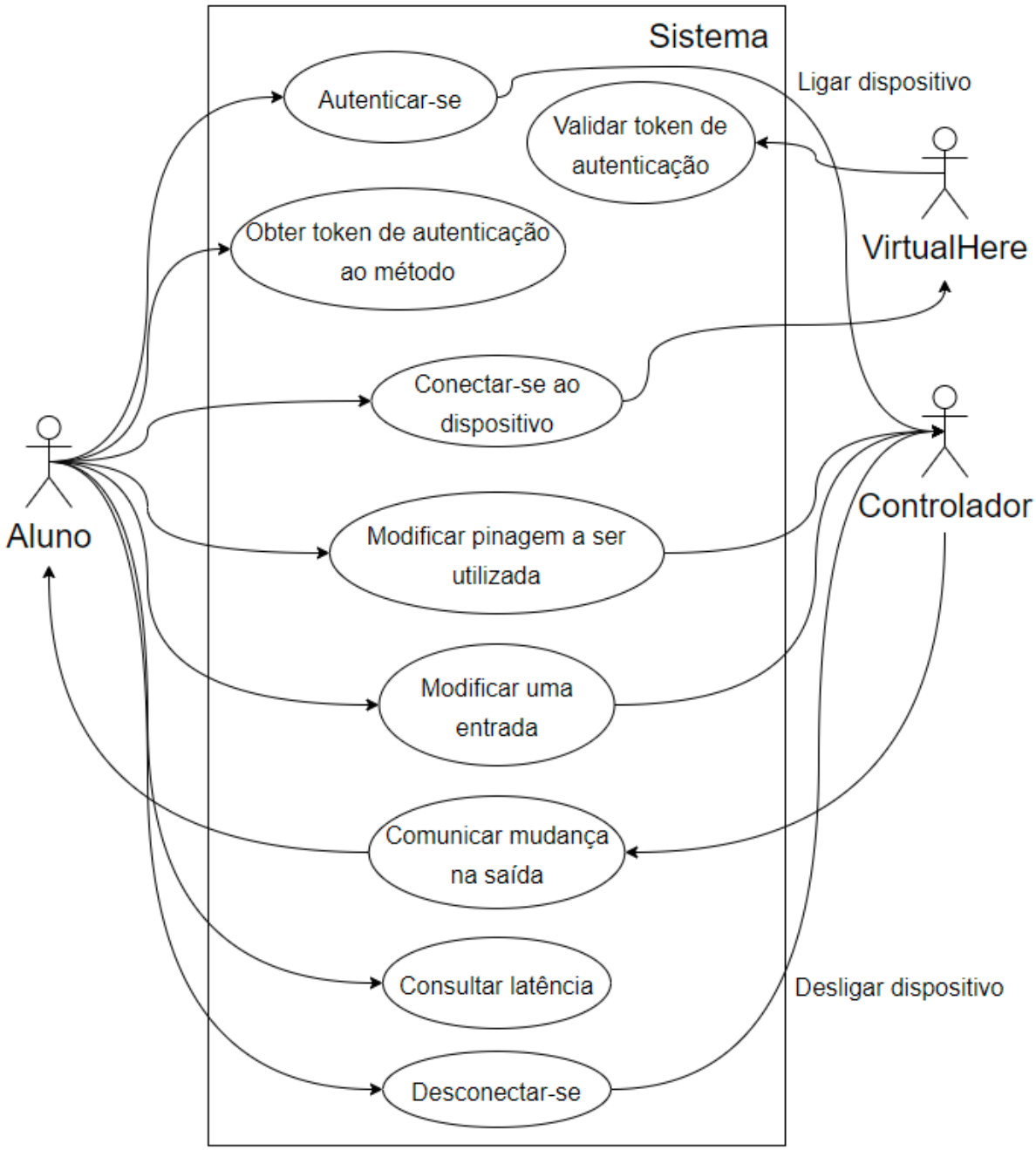
Outro modo de descrever o relacionamento dos sistemas é através de um diagrama de casos de uso que, embora dê uma visão simplificada, traz um panorama das funcionalidades disponíveis. Para não elevar a complexidade da explicação, não foi considerado o sistema de filas no diagrama de casos de uso presente na Figura 9.

Figura 8 – Escalabilidade



Fonte: DrawIO com imagens obtidas da aplicação Fritzing® e documentação oficial da Altera

Figura 9 – Diagrama de casos de uso



Fonte: Criado pelo autor através da ferramenta *DrawIO*

Um outro componente que não consta na figura é a base de dados, acessada pelo serviço *Web* (bloco 2) e pelo serviço de manipulação (bloco 4), e cuja principal função é guardar as informações referentes aos usuários e suas sessões de acesso, assim como guardar todas as configurações dos métodos, dispositivos e seus respectivos pinos.

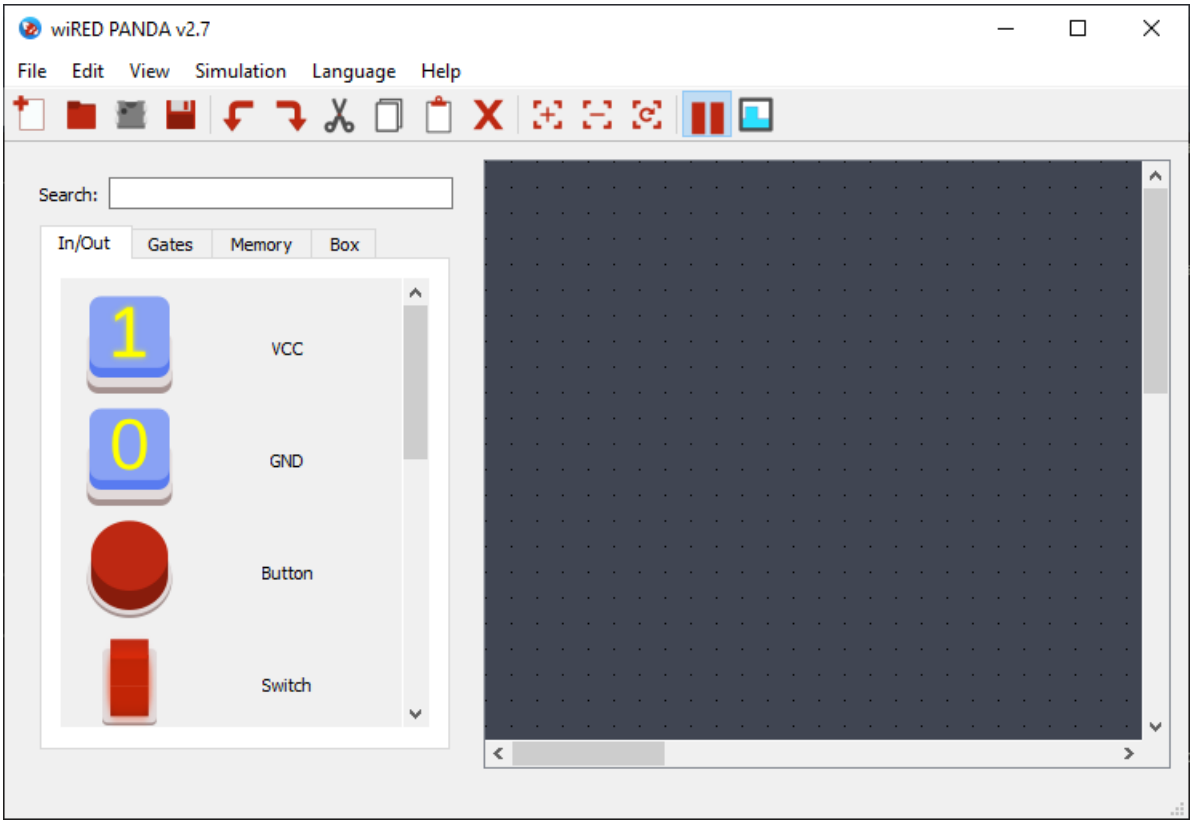
Com este descritivo, é possível melhor compreender o funcionamento do sistema completo. Entretanto, antes de adentrar em um maior detalhamento, é necessário tratar da escalabilidade do sistema (Figura 8), onde aqui entendemos escalabilidade como a

possibilidade de gerenciamento de múltiplos kits de forma simultânea, que é um dos pontos avaliados nos objetivos específicos deste trabalho e que deve ser garantida pela arquitetura do sistema.

Como não é exigido do *Raspberry PI* nenhuma entrada do GPIO, e sendo a única exigência portas *USB* (em pares), então esta abordagem pode suportar mais de uma placa. Contudo vamos avaliar alguns problemas na seção 4.5 devido ao elevado uso energético do controlador e possíveis formas de contornar esta limitação, fatores como este contribuirão para a limitação da quantidade de dispositivos.

4.2 Simulador de circuitos digitais - *WiredPanda*

Figura 10 – WiRED Panda



Fonte: Aplicação *WiRED Panda* operando no sistema *Microsoft Windows*

Sem perder as características principais do simulador de circuitos lógicos *WiRED Panda*, o laboratório remoto foi adicionado como uma funcionalidade adicional do simulador, apresentada como um novo componente a ser escolhido dentre os padrões de entrada e saída. Este foi chamado de *Remote Device* e possui uma seção de configuração diferenciada dos demais componentes, uma janela específica que alterna entre diferentes estados de acordo com a situação.

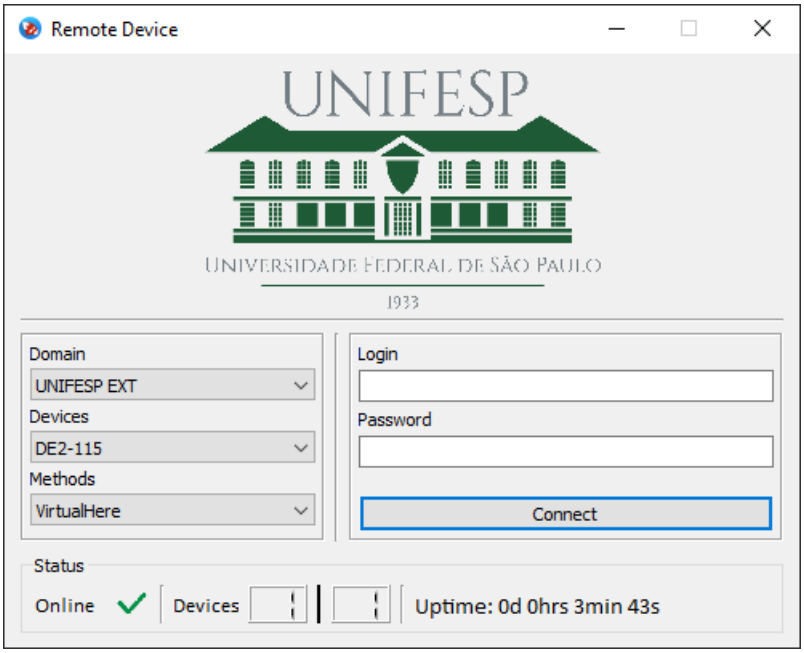
Para descrever o desenvolvimento das funcionalidades, serão exibidas as janelas do sistema comentando o contexto e as integrações, desta forma torna-se mais visível o processo.

O componente *Remote Device*, quando instanciado (arrastado para a área de trabalho do simulador), poderá ser configurado para operar de forma remota. Para isso, será primeiramente necessário fazer autenticação com algum serviço do laboratório remoto. Ao realizar a autenticação já deverá ser escolhido o dispositivo e o método de acesso desejado, pois o sistema foi previamente idealizado para operar com outros tipos de dispositivos, e o método de acesso *VirtualHere* poderia, por exemplo, ser substituído por outra solução.

4.2.0.1 Janela de autenticação

A primeira janela a ser discutida (Figura 11) será a primeira exibida cronologicamente, afinal, para as demais funções é necessário que o usuário tenha ao menos uma sessão inicializada.

Figura 11 – Janela de autenticação



Fonte: Aplicação *WiRED Panda*

No momento em que o usuário abre esta janela o sistema irá avaliar o conteúdo do arquivo *remotelab.xml* (Código 1) presente no diretório raiz do simulador *WiRED Panda*. Este arquivo basicamente é responsável por apontar, o domínio, a *URL* da *API* correspondente, e também informações referentes ao método de *hashing* das senhas inseridas no sistema. No caso deste trabalho, foi utilizado o algoritmo de sintetização de mensagem *MD5* visto que o serviço *LDAP* também trabalha com este mesmo algoritmo e isto

facilitaria a integração. Contudo atualmente o sistema conta com compatibilidade para realizar este *hash* utilizando outros algoritmos também, vide Tabela 3.

Tabela 3 – Opções *hashing* suportadas

Tipos
PLAIN
MD5
SHA-1
SHA-256

Fonte: O Autor

OBS: A opção *PLAIN* desativa o uso de qualquer algoritmo, contudo seu uso não é recomendado para fins diferentes de testes e validações.

```
<endpoints>
   <option name="UNIFESP">
      <url>http://remotelab.unifesp.br:8081/api/</url>
    <auth>MD5</auth>
   </option>
</endpoints>

```

Código 1 – Arquivo *remotelab.xml*

Após fazer a avaliação dos domínios, o sistema enumera as opções no primeiro seletor *Domain*, a primeira seleção será a carregada inicialmente. O sistema então envia uma solicitação *HTTP* do tipo *GET*, recebendo uma resposta em formato *JSON* do servidor *WEB*. Mediante a resposta é que a tela tem suas informações preenchidas, entre os dados apresentados estão a disponibilidade do sistema, o tempo que está em operação, e a lista de dispositivos e métodos correspondentes.

O logo também é carregado de forma similar, mas desta vez em outro caminho. Desta forma a aplicação fica neutra em termos de domínio, sendo passível de ser utilizada por outras Universidades sem a necessidade de alteração no código fonte do simulador.

Há outras requisições que são realizadas de modo a atualizar as listas desta janela, e estas são feitas conforme o usuário altera as opções referentes aos dispositivos e/ou domínio.

O pressionar do botão *Connect* dará início a um fluxo de autenticação, para facilitar, este fluxo será dividido em etapas.

OBS: O termo cliente refere-se ao simulador que faz papel de solicitante na conexão.

- Primeiramente uma mensagem do tipo *POST* é enviada para o sistema com as credenciais do usuário.

- O servidor *WEB* irá então avaliar se os dados de usuário e senha conferem, em caso negativo, este irá já retornar uma mensagem de erro coerente.

- Caso confiram as credenciais, então o sistema *WEB* irá criar um *token* de sessão do usuário, o qual chamaremos de chave secreta para fins deste trabalho. Após a criação e persistência na base de dados, a chave secreta é retornada para o cliente.

- Com esta chave secreta em mãos, o cliente irá estabelecer uma conexão persistente com um outro serviço. Este seria o serviço de manipulação de entrada e saída, contudo veremos que ele é responsável por outras funcionalidade que serão discutidas no decorrer deste trabalho.

- A conexão só será atendida caso a chave secreta do usuário confira com a sessão presente na base de dados, e a este ponto, o servidor já deverá retornar se foi possível achar uma vaga ao usuário ou se será necessário ficar na fila por um período.

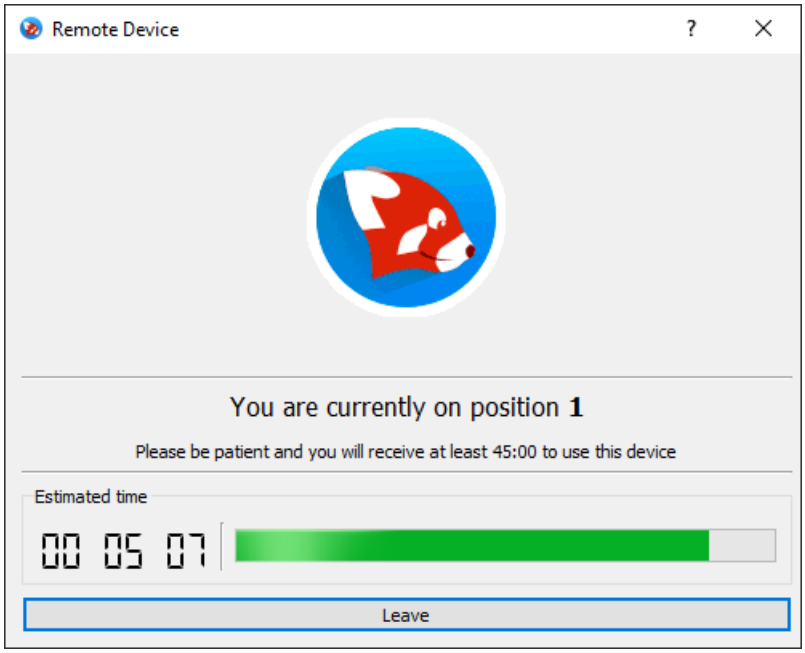
- Avaliando a situação quando há dispositivos livres, os dados referentes a conexão são passados neste momento e o usuário é levado para a nova tela de configurações avaliado na subseção 4.2.2. Caso não tenham dispositivos livres, então o cliente será indagado se deseja entrar na fila para utilizar o sistema.

É importante ressaltar que, neste momento a aplicação estará entrando em contato com dois servidores diferentes, um servidor *WEB* e o servidor de conexão persistente *TCP/IP*. O motivo pelo qual isto é feito é devido a necessidade, a partir deste ponto, de avaliar a presença do usuário, seja a presença dele na fila (caso sua conexão caia ele perderá seu lugar) ou de usuário ativo fazendo uso do sistema.

4.2.1 Janela da fila

Após o usuário ser questionado referente a possibilidade de entrar na fila, caso o usuário aceite enfrentá-la, este será levado a uma nova janela Figura 12 que irá trazer algumas informações para mantê-lo confortável durante sua espera. Caso o usuário não aceite, ele simplesmente irá desconectar.

Figura 12 – Janela de espera



Fonte: Aplicação *WiRED Panda*

Há três informações fundamentais que compõe esta janela. A primeira delas é a posição atual que o usuário se encontra; a segunda está escrita na mensagem com fonte de tamanho menor que diz o tempo que o usuário poderá utilizar o sistema quando este finalizar sua espera. Já a terceira e última informação, que está representada de duas diferentes maneiras por motivos estéticos, é o tempo estimado de espera. Este tempo será atualizado a todo segundo, e conforme a contagem aproxima-se de zero, a barra de progresso aproxima-se do fim.

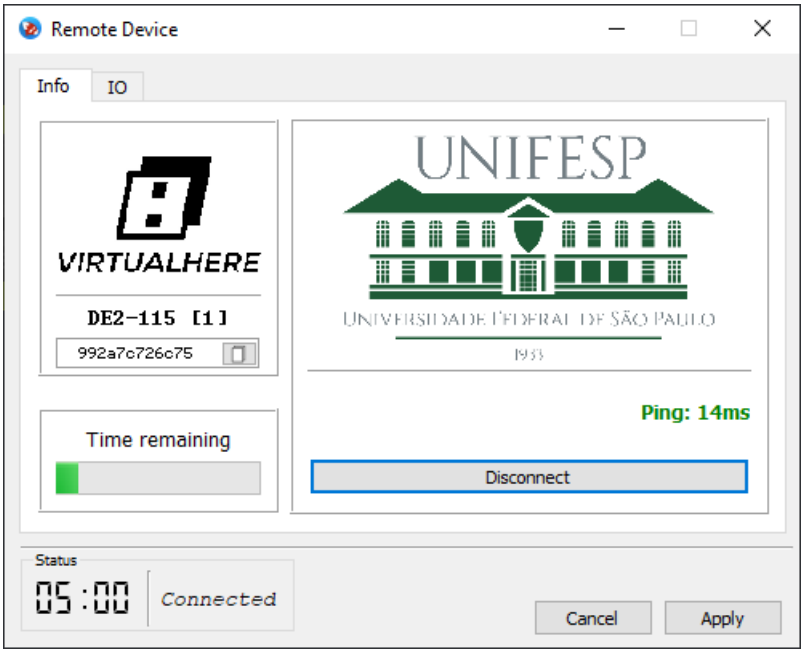
Como a conexão é persistente, então existem trocas de mensagens constantes que sincronizam o tempo estimado, ou seja, caso uma pessoa saia da fila, o seu tempo estimado irá cair pelo menos 45 minutos (que está como tempo de uso).

Para deixar a espera ainda mais confortável, o ícone do simulador *WiRED Panda* põem-se a girar passando a impressão que o sistema está carregando. Vale lembrar também que o usuário poderá sair da fila, caso queira, através do botão *Leave*.

4.2.2 Janela de configurações

Seja após o usuário enfrentar uma fila, ou ter o acesso garantido de forma imediata, a próxima tela que o usuário verá será a de configurações do dispositivo presente na Figura 13. Afinal, neste momento o usuário já se autenticou, já abriu uma conexão persistente com o servidor de gerenciamento de entrada e saída e agora pode utilizar o dispositivo.

Figura 13 – Janela de configurações do dispositivo



Fonte: Aplicação *WiRED Panda*

Na primeira página - página de informações - temos um logo referente ao método de conexão (logo este carregado via servidor *WEB*) e temos também novamente o logo referente ao domínio conectado (neste caso está presente novamente apenas por questões de estética e *design*).

Outras informações relevantes presentes nesta página são, o *token* de autenticação do método, o nome do dispositivo, o tempo restante para uso do dispositivo (em segundos e através de barra de progresso), a qualidade da conexão e a opção de desconectar.

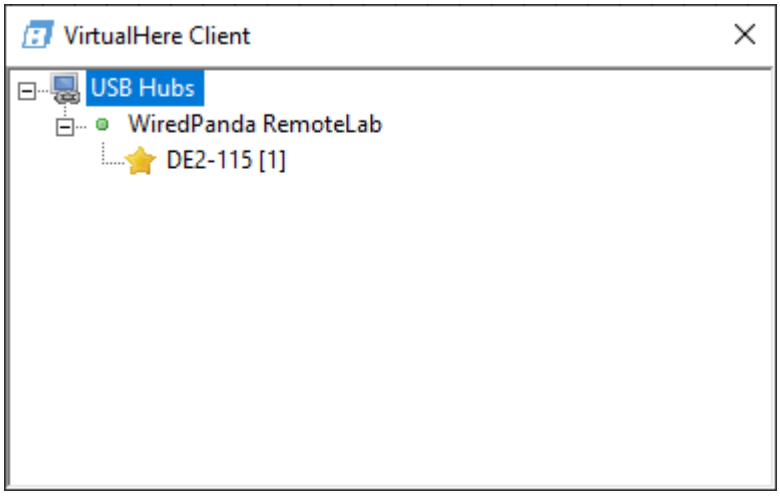
Antes de dar continuidade, devemos abrir neste momento uma pequena discussão para avaliar o motivo da implementação do método através de *tokens*. Existem dois casos de uso onde a pessoa que controla a entrada e saída de um dispositivo não será responsável por programá-lo. No contexto de uma avaliação de um trabalho, o docente pode, por exemplo, fazer uso de um projeto *WiRED Panda* descrito por um arquivo de extensão *panda* (responsável por salvar a mesa de trabalho dos componentes) e realizar a solicitação de um dispositivo para uso. Entretanto, neste caso específico o docente poderia solicitar ao aluno que programe a placa com o exercício pedido em sala de aula para que ele possa avalia-lo. O outro caso de uso em que este método de conexão por *tokens* irá ajudar, se dá na última disciplina da sequência de laboratórios de sistemas computacionais, o Laboratório de Comunicação Digital. Neste laboratório é solicitado aos alunos que conectem seus sistemas computacionais (desenvolvidos previamente) com os sistemas computacionais dos colegas, e para este caso, é possível que dois alunos (ou mais) trabalhem juntos para conseguir enviar dados de um sistema computacional para outro. Em ambos os casos, o

sistema de autenticação do método de conexão com a placa por *tokens* irá ajudar, afinal, não irá requerer que o aluno divida sua senha para que a pessoa possa programar a placa, será apenas necessário que o aluno passe o *token* que é gerado a cada nova conexão.

Para fazer uso do *token* o aluno poderá utilizar o botão disponível no canto direito da seção onde o *token* está descrito na janela mostrada na Figura 13 para fazer a cópia para a área de transferência, de modo a tornar mais simples a conexão no *VirtualHere*, que no caso, terá seu cliente aberto junto com esta janela (devido a uma integração que abre o executável desta aplicação). Como o *VirtualHere* possui uma opção para manter a janela sempre no topo (configurável por meio de um arquivo de configurações), a abertura da aplicação torna-se facilmente compreendida pelo usuário.

A janela do *VirtualHere* pode ser vista na Figura 14. Nela é possível avaliar a presença do kit de desenvolvimento *DE2-115*. O nome presente no *VirtualHere* é o mesmo presente na janela de configurações, isto para que fique intuitivo e fácil de reconhecer o dispositivo correto caso exista mais de um.

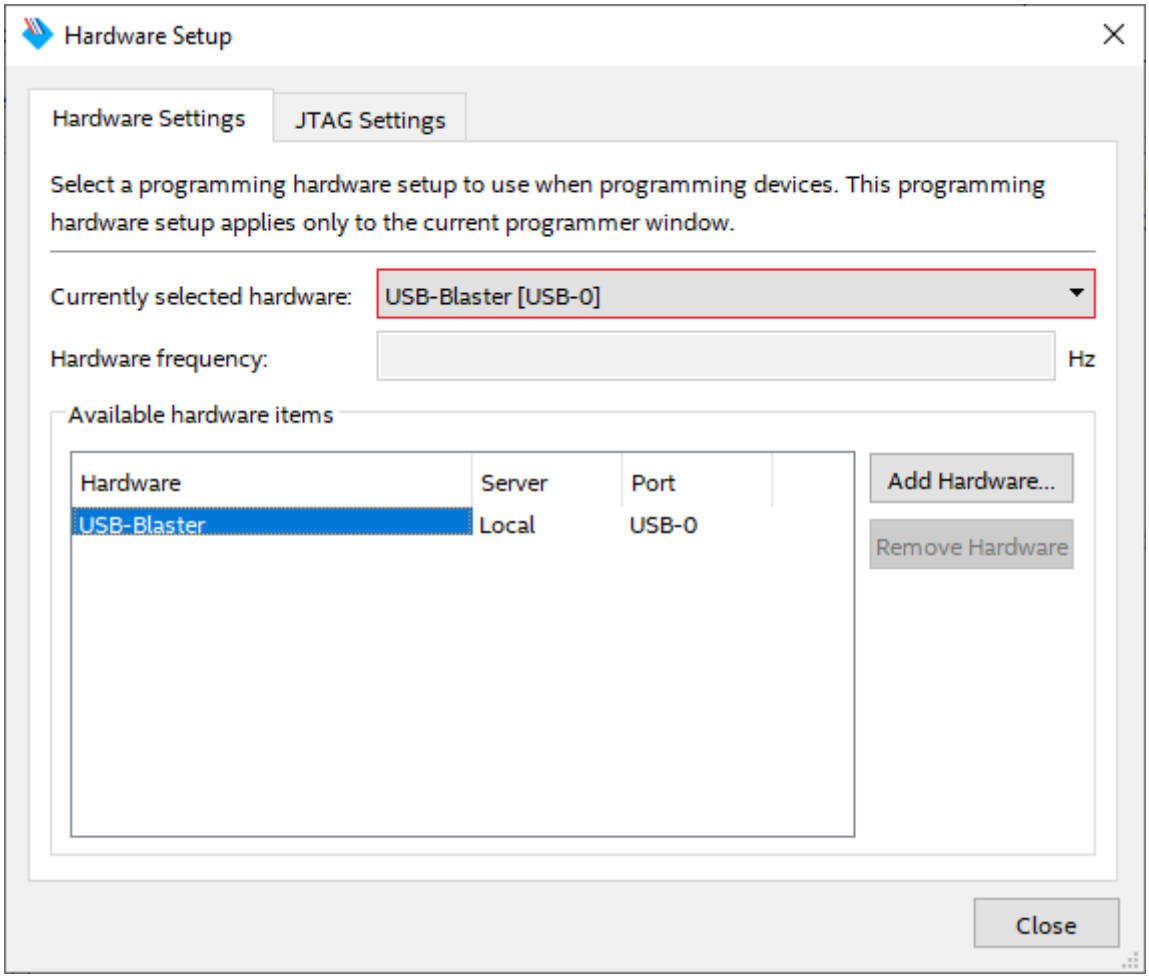
Figura 14 – Janela de configurações do dispositivo



Fonte: Aplicação *VirtualHere* desenvolvida por *VirtualHere Pty.*

Ao tentar fazer a conexão, a aplicação do *VirtualHere* irá solicitar uma senha, que neste caso é nada mais que o *token* da janela de configurações. Ao finalizar a conexão na aplicação *VirtualHere*, o dispositivo já será ligado de forma remota na máquina do usuário, podendo então já vê-la, por exemplo, na aplicação *Quartus* como mostra a Figura 15.

Figura 15 – Janela de seleção de dispositivo de *hardware*

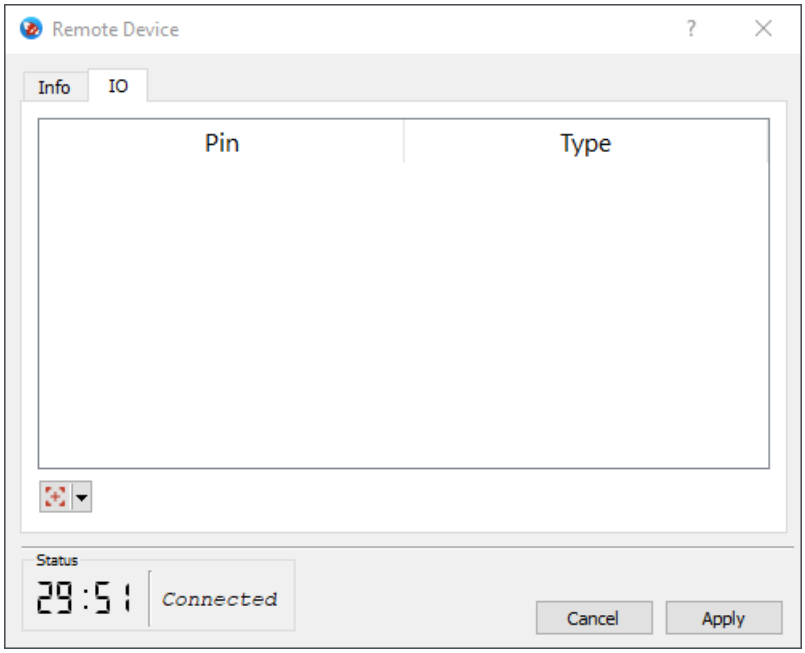


Fonte: Aplicação *Quartus Prime Lite Edition*

Não houve necessidade de nenhuma adaptação do *software Quartus* para o funcionamento desta integração, isto porque o *driver* do protocolo *USB/IP* deixa isto transparente, exatamente como se o dispositivo *USB* estivesse ligado a máquina do usuário.

Para finalizar, a última funcionalidade da janela de configurações do dispositivo esta presente na outra página, a página de pinos de entrada e saída avaliada na Figura 16. É nesta página em que o usuário irá cadastrar os pinos do dispositivo a serem utilizados, e devido a prévia configuração deste dispositivo na base de dados, é possível avaliar a lista de pinos disponíveis já na aplicação (Figura 17).

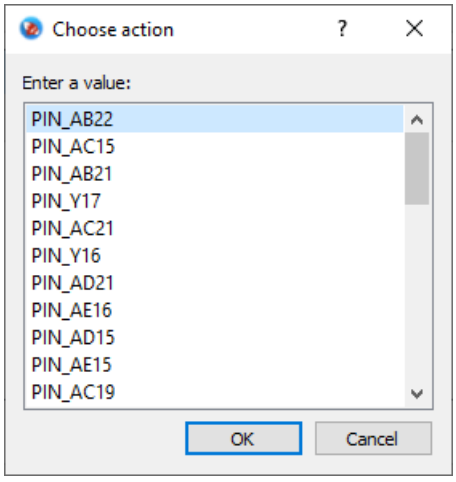
Figura 16 – Página de configurações de pinos



Fonte: Aplicação *WiRED Panda*

Vale lembrar também que todos os pinos deverão ser descritos como um pino de entrada, ou saída. Dependendo a implementação (a nível de *hardware*) alguns pinos podem somente ser compatíveis, por exemplo, com entrada de dados, outros somente com saída de dados. Para esta aplicação isto não é um problema visto que ela irá identificar que o usuário não pode utilizar determinado pino como entrada, ou saída (também de acordo com prévia configuração na base de dados).

Figura 17 – Pinos disponíveis para seleção

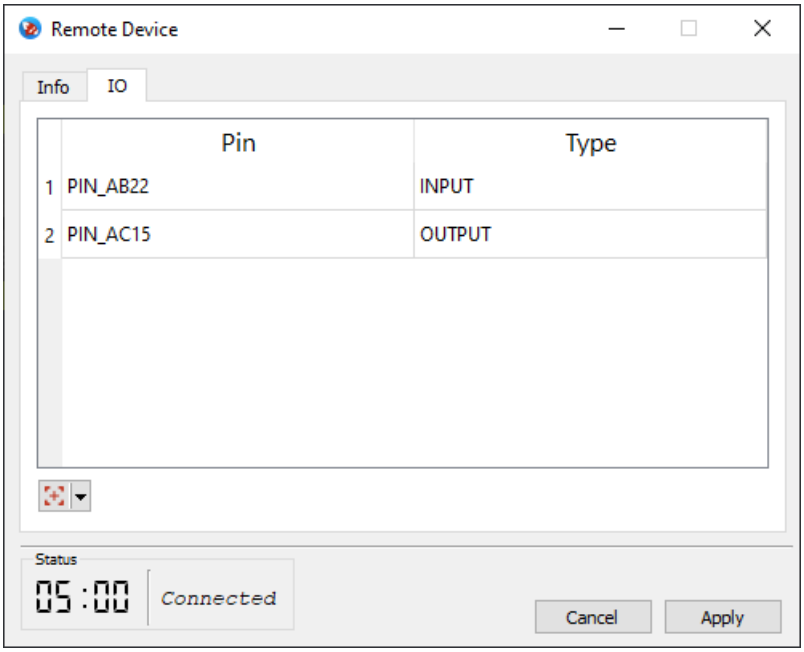


Fonte: Aplicação *WiRED Panda*

Ao término da configuração, será possível obter uma lista como a indicada na

Figura 18.

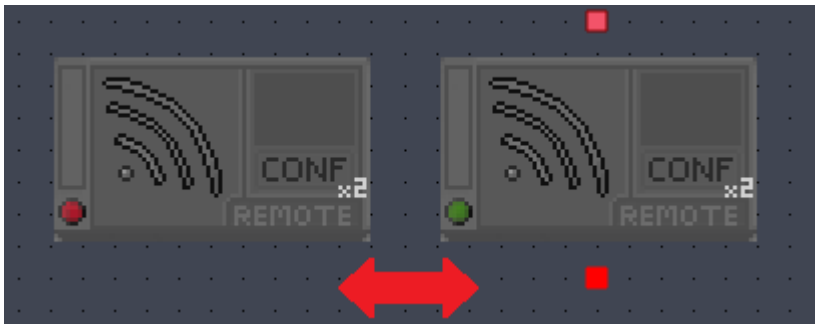
Figura 18 – Página de configurações de pinos



Fonte: Aplicação *WiRED Panda*

Ao aplicar as alterações, o componente *Remote Device* presente na mesa de trabalho do simulador irá agora ser reconfigurado para apresentar as saídas e entradas configuradas, como mostra a Figura 19.

Figura 19 – Página de configurações de pinos

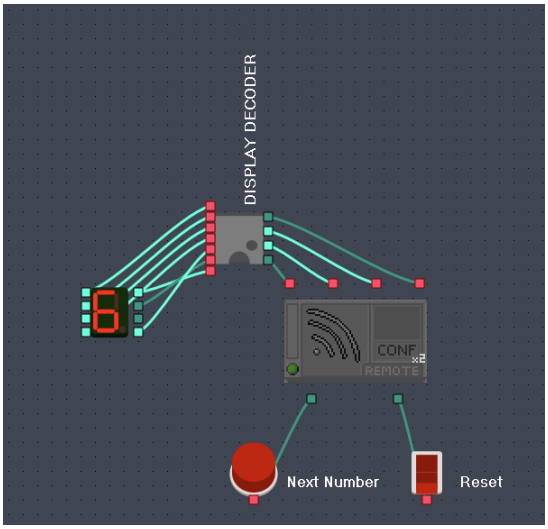


Fonte: Aplicação *WiRED Panda*

Desta forma, é possível conectar os componentes do simulador para utilizá-los como interface de entrada e saída. Aqui coloca-se a seguinte questão, afinal seria mesmo esta abordagem somente um laboratório remoto? Uma vez que, está sendo feita aqui uma integração com um simulador que se encaixaria na descrição de um laboratório virtual. Deixo esta questão à reflexão dos leitores e voltaremos a discuti-la no Capítulo 6.

Na Figura 20 é mostrado um exemplo de circuito que faz uso tanto do kit de desenvolvimento com *FPGA*, assim como de circuitos simulados dentro do *WiRED Panda*, mostrando possibilidade de fazer uso de uma solução híbrida, parte desempenhada pelo componente *FPGA* e parte simulada.

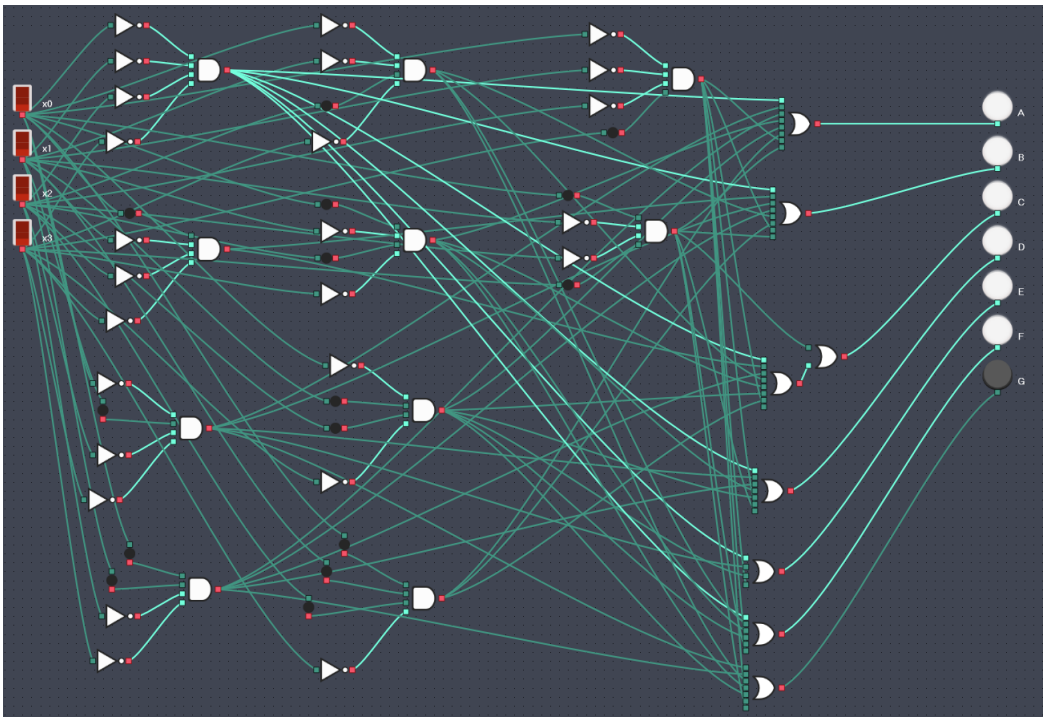
Figura 20 – Exemplo de circuito contador



Fonte: Aplicação *WiRED Panda*

OBS: A caixa preta escrita *Display Decoder* é o circuito presente na Figura 21.

Figura 21 – Decodificador para *display* de 7 segmentos



Fonte: Aplicação *WiRED Panda*

O código *Verilog HDL* utilizado encontra-se em Código 2, contudo, de modo a simplificar, neste código não está implementado um circuito *debouncer*.

```
1  module ContadorCicl(
2          input in,
3          input reset,
4          output reg [3:0]out
5  );
6
7  always@(negedge in) begin
8
9          if (reset == 1'b0) begin
10                 if(out[3:0] == 4'b1001)
11                         out[3:0] = 4'b0000;
12                 else
13                         out[3:0] = out[3:0] + 4'b0001;
14         end else
15                 out[3:0] = 4'b0000;
16
17 end
18
19 endmodule
```

Código 2 – Contador cíclico - sem *debouncer*

4.3 Servidor

Foi escolhido o microcomputador *Raspberry PI 4* devido seu baixo custo e seu tamanho reduzido, capaz de atender as demandas deste projeto. Foi utilizado o sistema operacional *Raspberry PI OS* (previamente chamado de *Raspbian*) que é o padrão destes microcomputadores devida sua alta compatibilidade e otimizações para funcionar em *hardware* de arquitetura *ARMv8*. Este sistema é uma variante do *Debian*, e possui uma grande quantidade de pacotes (superior a 30 mil) compatíveis, sendo muito similar ao manuseio de sistemas *Linux* como o *Ubuntu*, *Mint* e o próprio *Debian* (FOUNDATION, (acessado em 22 de fevereiro de 2021)).

O acesso pode ser feito diretamente ao computador ligando-o ao monitor com um teclado e *mouse* para fazer o controle, contudo, de modo a agilizar o uso, recomenda-se o uso de *SSH* para fazer controle do dispositivo, sendo necessário ao menos um pouco de experiência com o *Terminal* de sistemas *Linux*.

Os pacotes que serão necessários para este projeto já encontram-se instalados, sendo interessante somente fazer a atualização do *Python* para uma versão mais recente caso

seja possível.

Para gerenciar e orquestrar os diferentes serviços, utiliza-se o *Systemd* também disponível previamente neste sistema. Seu uso é importante devido a facilidade de trabalhar com os serviços (consultar *logs*, configurar inicialização automatizada e fácil reinicialização por linha de comando).

O presente trabalho não tem o foco de dar instruções em como replicar o ambiente criado, mas sim de explicar os elementos utilizados, a arquitetura utilizada e descrever os principais componentes. As instruções para a instalação deste serviço estarão disponíveis na página da ferramenta *Github*.

4.3.1 Serviço *WEB*

O primeiro serviço a ser tratado é o servidor *Web*, responsável pela autenticação dos usuários, criação das sessões, e fornecimento das imagens e arquivos. A ideia de existir um serviço *Web* são duas. A primeira é para segregar as responsabilidades, desta forma requisições que não envolvam a necessidade de alta responsividade podem ser feita através do protocolo *HTTP* além disso, integrações *HTTP* são mais fáceis de se fazer, seja com sistemas *LDAP* ou, como será utilizado, com o *VirtualHere*.

Para o desenvolvimento foi utilizado somente a linguagem *Python v3.7.3* e bibliotecas de desenvolvimento *Web* - neste caso *Flask*.

Devido a necessidade de um tutorial para ensinar os alunos a utilizarem o serviço, além da *API*, uma página simples foi adicionada ao sistema, esta utiliza *Bootstrap* e *JavaScript*, e seu conteúdo é lido de arquivos *Markdown* (uma linguagem simples de marcação) que facilita a edição do conteúdo destas páginas. O uso de *Markdown* na aplicação foi possível através do uso da dependência *Flask-Markdown* que torna simples sua integração.

Após finalizado, foi utilizado o *Waitress*, um *WSGI (Web server gateway interface)* capaz de disponibilizar a aplicação com qualidade e segurança de ambiente produtivo.

4.3.1.1 Rotas

As rotas principais para o sistema foram mapeadas em */api* e estas são necessárias para disponibilizar à aplicação um método de autenticação, validação de *token* do método e obtenção das imagens do sistema. As principais rotas foram descritas na Tabela 4.

Tabela 4 – Rotas do serviço *WEB*

Rotas	Método	Descrição
/api	GET	Retorna informações gerais sobre o servidor
/api/logo	GET	Retorna o logo do domínio
/api/method	POST	Retorna a imagem do método de acordo com a sessão
/api/auth_device	POST	Faz a validação do token do método
/api/login	POST	Faz a autenticação do usuário e gera a sessão

Fonte: O Autor

Outras rotas foram criadas para a página informativa (que contém o tutorial) e para a página de administração do serviço, contudo, não são o foco deste trabalho.

4.3.1.2 Conexão com base de dados

Para esta solução, foi utilizada uma base de dados *MySQL - MariaDB*, esta base é responsável por manter informações referentes aos usuários (inclusive aqueles vindos de integrações com *LDAP*), assim como as informações referentes aos dispositivos configurados.

Para fazer a conexão com esta base de dados foi utilizado o pacote *mysql-connector-python*, e com ela foi escrita uma classe *DBService* e adicionada como módulo da aplicação.

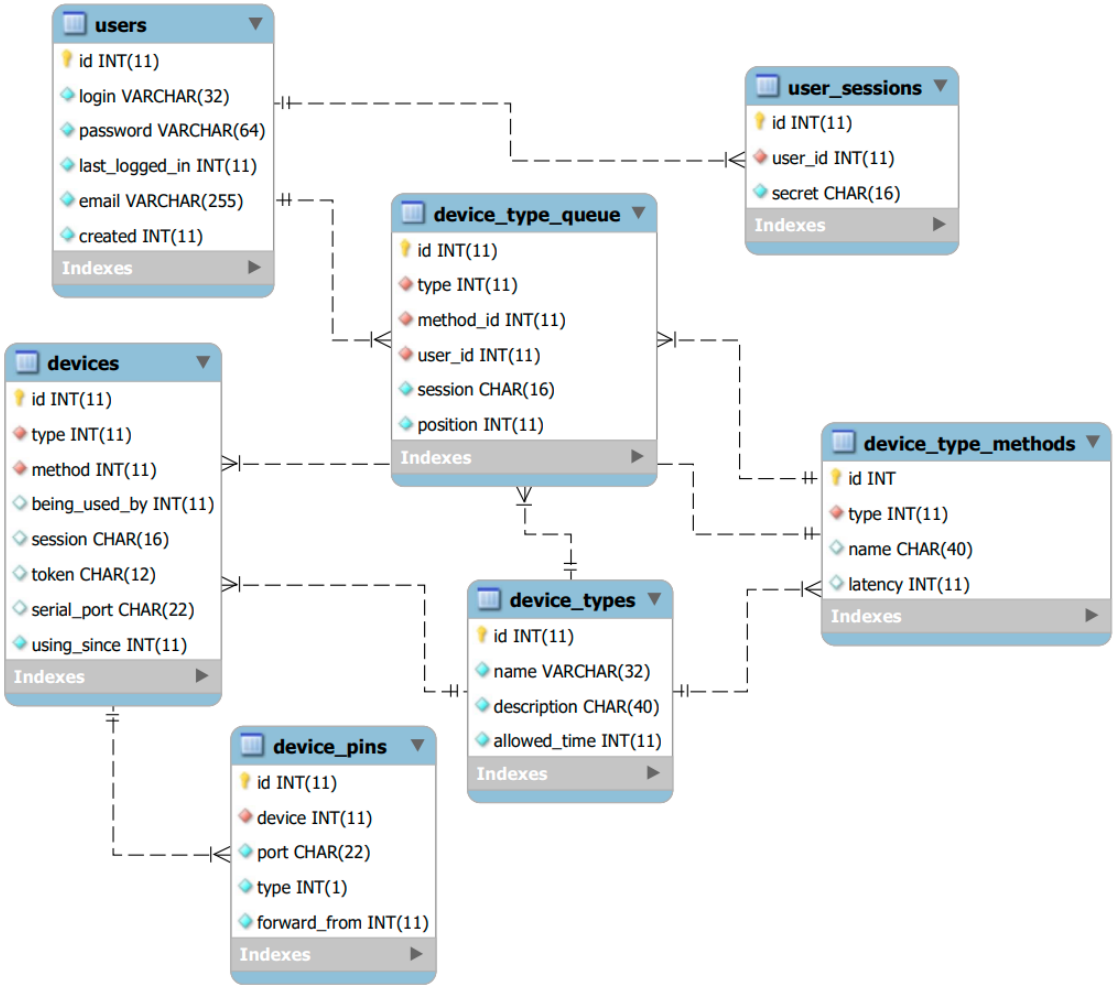
Para facilitar o desenvolvimento, uma integração com arquivos *SQL* foi realizada de modo que a própria aplicação sobe a estrutura da base de dados no ambiente de desenvolvimento tornando simples o processo de mudanças na estrutura durante os testes.

4.3.2 Base de dados

A base de dados *MySQL - MariaDB* foi utilizada de modo a guardar informações referentes aos usuários e também para guardar as informações referentes as configurações do sistema.

A estrutura da base é bastante simples, e possui um total de sete tabelas conforme o diagrama de entidade e relacionamento presente na Figura 22.

Figura 22 – Diagrama de entidade e relacionamento



Fonte: Diagrama desenhado no *MySQL Workbench* seguindo o modelo proposto por James Martin (MARTIN, 1989)

Embora seja auto-explicativa a estrutura da base de dados, o propósito de cada tabela será comentado a seguir.

- *users*: esta tabela possui as informações do usuário. Para os casos onde o usuário foi sincronizado do *LDAP*, o campo *password* recebe o valor *LDAP*;

- *user_sessions*: cada usuário poderá ter diversas sessões abertas simultaneamente, algo que permite, por exemplo, a utilização de múltiplos dispositivos;

- *device_types*: é nesta tabela que fica cadastrado os tipos de dispositivos controlados pelo sistema, isto uma vez que o sistema pode atuar com mais de um tipo de dispositivo, seja um outro kit de desenvolvimento com *FPGA*, ou uma placa de propósito completamente diferente (por exemplo, um PIC 18F4550);

- *device_type_methods*: para cada tipo de dispositivo um método diferente é cadastrado, isto porque a atuação do método pode ser diferente para cada tipo de dispositivo;
- *devices*: para cada placa de um determinado tipo que estiver disponível para uso, um novo item nesta tabela deverá existir. Muitos atributos são alterados no momento de uso do dispositivo, tal como o usuário que está utilizando, a sessão e assim por diante;
- *device_type_queue*: esta tabela contém os usuário que estão enfrentando fila para utilizar o sistema, as posições na fila são coordenadas por aqui;

Cada dispositivo configurado deve estar associado a somente um método, ou seja, supondo que existam dois métodos de utilização, deverão existir dispositivos segregados para cada método. Esta implementação talvez seja alterada no futuro se for possível, por exemplo, utilizar múltiplos métodos para um mesmo dispositivo físico.

4.3.3 Serviço de manipulação de entrada e saída

De modo a manter simples a instalação deste sistema, o serviço de manipulação de entrada e saída foi acoplado a solução *Python* anterior, embora isto prejudique a performance do sistema (como explicado a frente), tornou-se reutilizável toda a estrutura da regra de negócio necessária, assim como o serviço de conexão a base de dados *DBService*.

O desempenho passa a ser prejudicada devido ao fato de que o processo *Python* não trabalha de forma paralela, apenas concorrente. Deste modo a não segregação em dois processos (que poderiam trabalhar de forma paralela) irá resultar na queda de desempenho citada anteriormente, entretanto, supondo a não superutilização do núcleo de processamento, não há motivos suficientes para desmembra-los.

Este serviço apresenta um grau de complexidade bastante superior quando comparado com o serviço *WEB* da subseção 4.3.1, isto porque para operar com diversos clientes de forma concorrente deve-se trabalhar com segregação em *threads* e ser implementado um sistema de semáforos para controlar a sessão crítica no *BusinessService* (serviço de regra de negócio).

Para a implementação do servidor *TCP*, foi utilizado a biblioteca *socket* disponível no próprio *Python*. Já o controle concorrente dos clientes foi realizado por meio das bibliotecas *_thread* e *threading*, também disponíveis por padrão.

Outro serviço interno importante é o serviço de comunicação serial, este atua somente quando um cliente possui uma conexão ativa com algum dispositivo. A comunicação serial é uma forma rápida e fácil de comunicar-se com o microcontrolador *ATmega 2560*, e deste modo, é utilizada para fazer o controle dos pinos.

Poucos são os tipos de mensagens trocados entre a aplicação cliente (*WiRED Panda*) e a aplicação servidor, e sua estrutura, diferente do protocolo *HTTP*, é realizada com envios *byte* a *byte*, ao invés de mensagens textuais, possibilitando assim mensagens de tamanho reduzido e de forma a não estourar o *MTU* (*Maximum Transmission Unit*). Os tipos de mensagens e a estrutura será comentada com maior detalhe na seção 4.5.

A ideia principal por trás do uso de um servidor de conexão persistente para alta responsividade é o fato de que conexões persistentes *TCP* são bidirecionais, desta forma, no momento em que o cliente precisar notificar uma mudança dos pinos de entrada, este poderá simplesmente mandar um pacote para isto. Por serem bidirecionais, caso o servidor precise notificar a mudança no estado de um pino de saída, este poderá enviar o pacote para notificar de forma imediata também. No protocolo *HTTP* as conexões são unidirecionais, de modo que seria necessário ao cliente solicitar as respostas, o que além de exigir o envio de mensagens em uma alta taxa para o servidor, ainda iria contribuir para uma maior latência resultante.

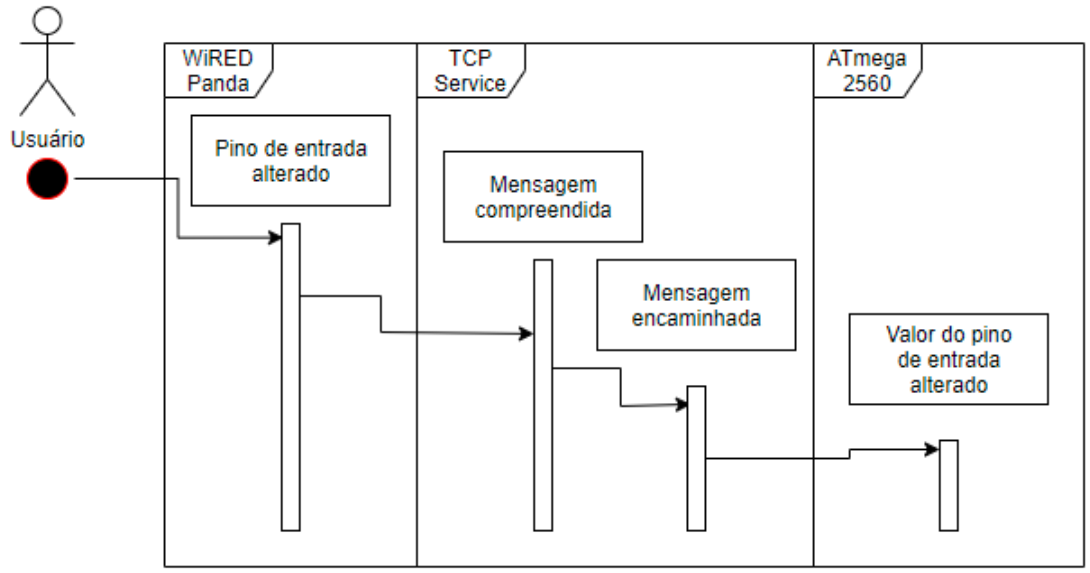
Para rastrear a conexão do usuário, e também para manter a conexão *TCP* aberta (sem que roteadores cortem a conexão por falta de comunicação), mensagens são trocadas seguindo um intervalo fixo - atualmente configurado de um segundo. Estas mensagens são importantes para avaliar caso uma conexão tenha caído, sendo possível inclusive implementar um sistema de recuperação de conexão (para caso o cliente volte momentos depois com inclusive um novo endereço *IP*).

Caso o usuário esteja enfrentando uma fila para utilizar o sistema, são estas mensagens trocadas, seguindo um intervalo fixo, que irão manter o usuário conectado com o servidor e também irá manter o tempo restante sincronizado.

No momento em que o usuário está fazendo uso do dispositivo, os pacotes que definem os estados dos pinos são simplesmente traduzidos e repassados para o alvo necessário, ou seja, quando o usuário mudar o estado de uma chave de entrada, o cliente *WiRED Panda* irá enviar uma notificação para este serviço que em seguida irá traduzir a mensagem e repassar para o microcontrolador através de uma conexão serial como apresentado na Figura 23.

A conexão serial com o microcontrolador por sua vez também é bidirecional, assim, quando for avaliado uma mudança no estado de uma chave de saída o microcontrolador irá notificar o servidor *TCP* que por sua vez irá redirecionar a mudança de estado por *TCP* para o cliente *WiRED Panda*.

Figura 23 – Diagrama de sequência



Fonte: Desenhado no *DrawIO*.

4.3.4 Serviço de concessão de acesso aos dispositivos *USB*

Para dar acesso aos dispositivos *USB* (kit de desenvolvimento *DE2-115*) foi utilizada a solução *VirtualHere*, a qual mostra-se bastante customizável e facilmente integrável ao projeto.

Para garantir acesso ao dispositivo somente para o aluno que está fazendo uso no momento, existe uma integração *HTTP* com o serviço *Web* que valida o *token* inserido pelo usuário, caso o *token* esteja correto, o *VirtualHere* vai então garantir o acesso.

Mediante a compra da licença para uso do *VirtualHere*, recomenda-se utilizar a aplicação servidor compilada especificamente para o *Raspberry PI 4*, visto que possui inúmeras otimizações e é a recomendada pelo equipe de desenvolvimento do produto.

Algumas configurações a serem tratadas nesta seção exigem o uso do *software* licenciado para funcionar, porém, é possível utilizar o serviço em uma versão de avaliação com algumas limitações, contudo as integrações não serão efetivas e versões otimizadas do *software* não funcionarão.

4.3.4.1 Configurações necessárias para controle de acesso na solução *VirtualHere*

O *VirtualHere* possui um sistema de controle de eventos que pode ser utilizado para habilitar ou desabilitar funcionalidades, assim como é possível utilizá-lo para configurar um fluxo de autenticação ao tentar fazer uso de algum dispositivo.

Primeiramente foi necessário garantir que o kit de desenvolvimento *DE2-115*

estivesse sempre com o apelido *DE2-115 [1]*, pois ele utiliza o apelido do dispositivo para fazer a autenticação no serviço *Web* através de uma requisição *POST* em */api/auth_device*. Sendo assim necessário trocar o apelido e em seguida desativar a opção de qualquer usuário poder fazer esta troca, para isto foi utilizado o sistema de controle de eventos.

A documentação presente no site do fornecedor foi suficiente para uma fácil compreensão da usabilidade, e muitos dos exemplos fornecidos faziam exatamente aquilo que era necessário. Para configurar um evento é necessário apontar um *script bash* a ser executa no momento que aquela ação ocorre. Dependendo do retorno do *script* a ação é efetivamente executada, ou pode ser desativada dependendo do resultado do *script*.

Na Tabela 5 estão presentes os eventos que foram utilizados para implementar esta solução, sendo a maioria deles apenas desativado para não resultarem em mudanças que um usuário comum não deveria poder realizar.

Tabela 5 – Eventos utilizados

Eventos	Comando	Descrição
onServerRename	exit 1	Previne usuário de mudar o nome do servidor
onChangeNickname	exit 1	Previne usuário de trocar apelido de algum dispositivo
onDeviceIgnore	exit 0	Previne usuário de adicionar dispositivo a lista de ignorados (o que fará com que o dispositivo desapareça)
onAddReverseClient	exit 0	Previne usuário de inserir um cliente de conexão reversa
onRemoveReverseClient	exit 0	Previne usuário de remover um cliente de conexão reversa
clientAuthorization	auth.sh	Realiza o fluxo de autenticação de usuário

Fonte: O Autor

O retorno de alguns eventos tem ações restritivas quando o *script* possui retorno igual a zero, outros quando tem o retorno igual a um. Portanto é importante consultar a documentação oficial para cada caso visto que não existe um padrão lógico.

No Código 3 é possível encontrar o conteúdo do *script auth.sh*.

```
#!/bin/sh

if [ "$6" = "" ]; then
  exit 2
fi

logger "[vhusbdarmpi4] Authorizing -> '$1' '$2' '$3' '$4' '$5' '$6' '$7'
    '$8' '$9'"

result=$(curl -s --data "name=$8&token=$6" http://127.0.0.1:8081/api/
   auth_device)

if [ "$result" = "Success" ]; then
  logger "[vhusbdarmpi4] Authorized!"
  exit 1
fi

logger "[vhusbdarmpi4] NOT authorized"

exit 0
```

Código 3 – *Script* auth.sh responsável pela autenticação do *token* inserido pelo usuário

Segundo a documentação, retorno igual a *dois* solicita a inserção de uma senha, a qual será repassada no parâmetro *$6* utilizando o algoritmo de *hashing MD5*. Retorno igual a *um* significa que o usuário poderá se autenticar (fazer uso do dispositivo), e retorno igual a *zero* irá ocasionar em erro. O parâmetro *$8* possui o apelido do dispositivo que o usuário deseja se conectar.

Em caso de necessidade de adicionar novos dispositivos, será necessário trocar os apelidos deles, há duas maneiras de se fazer isto:

- Desativar restrição de alterar o apelido, e em seguida pela janela do *VirtualHere* selecionar a opção *Rename...* e depois ativar novamente a restrição;
- Ou poderá avaliar as informações referentes ao dispositivo na janela do *VirtualHere* utilizando a opção *Properties* e preencher o parâmetro *DeviceNicknames* presente no arquivo principal de configuração do servidor *config.ini*;

Para qualquer uma das maneiras, será necessário reiniciar o serviço.

4.4 Interface elétrica

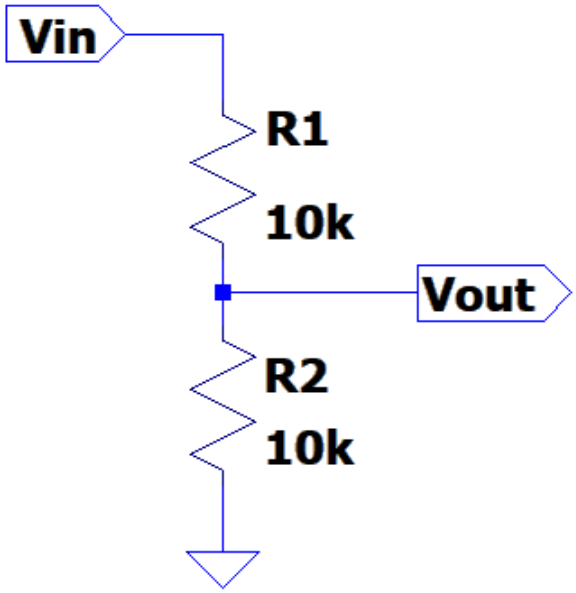
Para fazer a conexão entre o microcontrolador *ATmega 2560* e o kit de desenvolvimento *DE2-115* foi necessário torná-la compatível de um ponto de vista elétrico. Sendo o

microcontrolador um componente de comunicação *TTL* (*Transistor-transistor logic)* que opera na tensão de *5V*, para torná-lo coadunável com o kit de desenvolvimento *DE2-115*, também *TTL*, foi necessário realizar o ajuste da tensão para que este opere dentro dos valores de tensão recomendados pelo fabricante (ALTERA CORPORATION, 2016).

Foi utilizado um circuito divisor de tensão com resistores de *10k*Ω. Resistores de alta resistividade foram selecionados de modo a limitar a corrente drenada pelos componentes caso o aluno faça o mapeamento dos pinos incorretamente. Embora a tensão de operação da *GPIO* do kit de desenvolvimento *DE2-115* seja *3.3V*, ao dividir a tensão para *2.5V* o dispositivo ainda reconhecerá os diferentes níveis lógicos, então a fim de simplificar, a tensão foi dividida pela metade fazendo o uso de dois resistores iguais.

O divisor de tensão (presente na Figura 24), quando utilizado de forma inversa - onde a tensão é inserida em *Vout* ao invés de *Vin*, proporciona um resistor em série *R1* na linha que atua como um limitador de corrente. Assim, a tensão do kit de desenvolvimento não será dividida, possibilitando o reconhecimento dos níveis altos de tensão do kit de desenvolvimento pelo microcontrolador *ATmega 2560*, que no caso deverá ser superior a *3.0V* (ATMEL CORPORATION, 2014).

Figura 24 – Circuito divisor de tensão

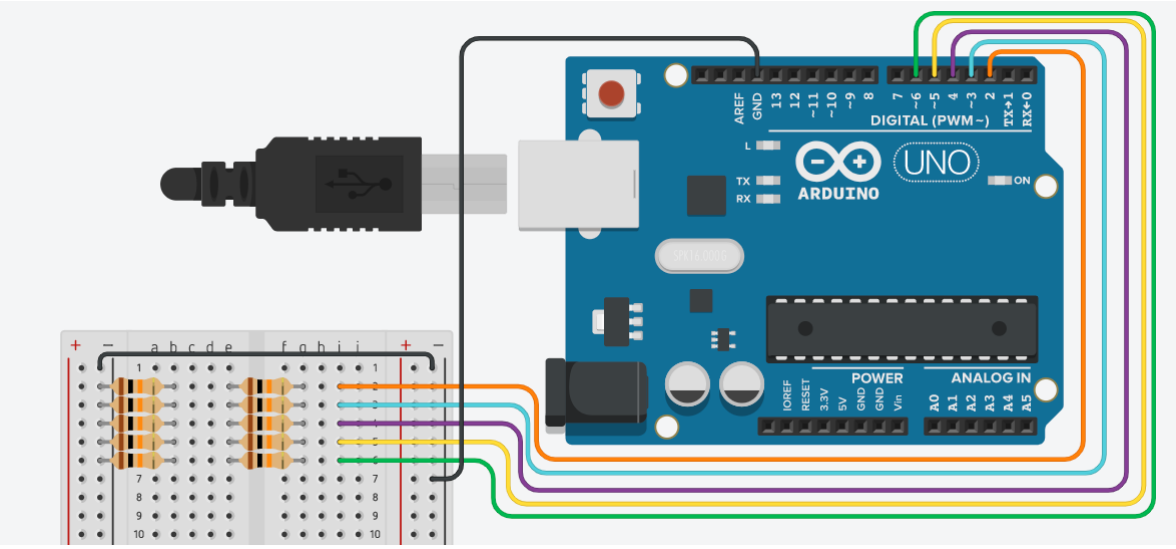


Fonte: Simulador de circuitos *LTSpice*

Devida sua simplicidade, o circuito foi montado utilizando uma *breadboard*, contudo, é possível fazer uso de circuitos impressos para reduzir esta interface elétrica, podendo estes possuírem também o microcontrolador *ATmega 2560* de modo a miniaturizar a solução e torná-la menos frágil.

A interface elétrica é similar a presente na Figura 25. Contudo será realizada para todos os 36 pinos de propósito geral ao invés de apenas cinco pinos como mostrado. Será entre os dois resistores que os pinos do kit de desenvolvimento *DE2-115* deverá ser conectado.

Figura 25 – Exemplo de ligações

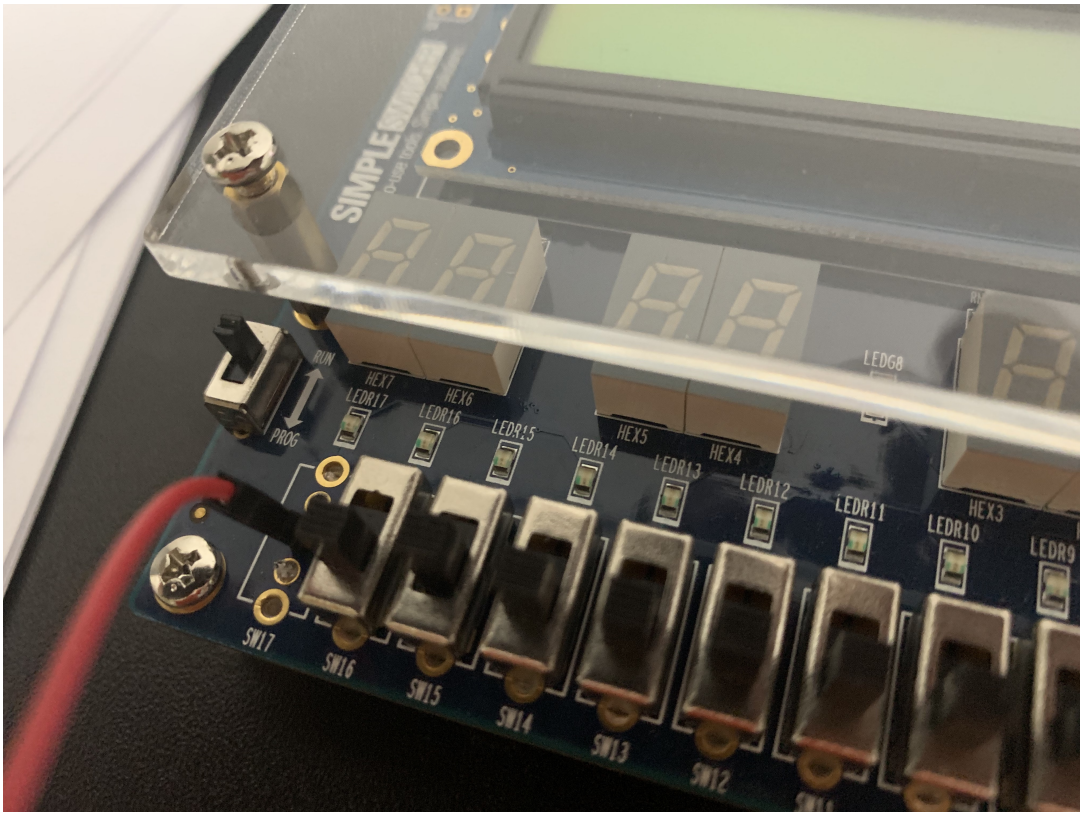


Fonte: Simulador de circuitos *Tinkercad*

Além dos pinos de propósito geral, foi ligado também um pino de uma chave que estava com defeito através de mais um divisor de tensão igual aos avaliados acima. Para isto, foram realizados testes para compreender o funcionamento das chaves e foi identificado que estas atuam sobre uma tensão de $2.5V$ (diferentemente da tensão do *GPIO* de $3.3V$) sendo essa tensão controlada pelo *jumper JP7* do kit de desenvolvimento *DE2-115*.

Como a chave selecionava entre um valor alto de $2.5V$ e um valor baixo *GND*, não seria possível fazer a ligação sem antes fazer a remoção dela, visto que sem este procedimento, é possível que um curto-circuito pudesse ocorrer entre a tensão de saída do microcontrolador e a tensão referência do kit (*GND*).

Figura 26 – Remoção da chave

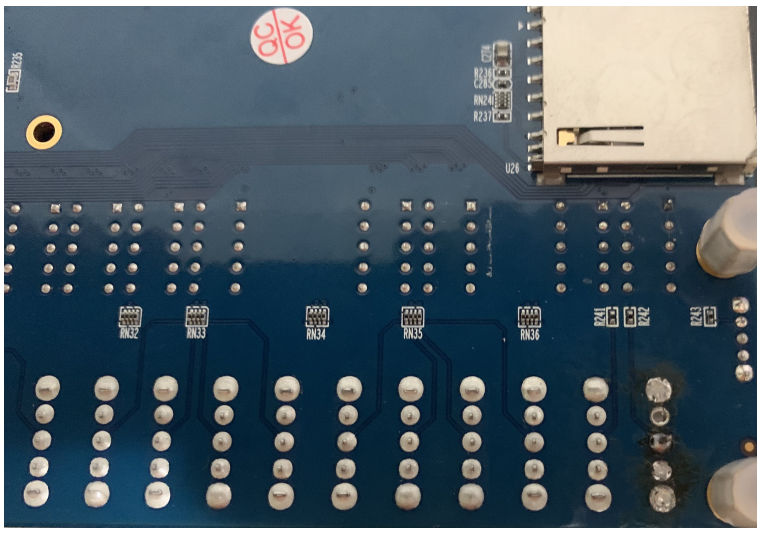


Fonte: O Autor

Para retirar a chave, foi utilizado um soldador e um sugador de solda. Após retirá-la e realizar a ligação de um *jumper* no local (Figura 26), este foi conectado entre os dois resistores do circuito divisor de tensão discutido anteriormente. Desta forma, caso seja feito o uso desta chave, então, a interface elétrica irá regular a tensão de saída de 5V do microcontrolador para a tensão de 2.5V - mesma tensão antes selecionada pela chave.

Embora a solução tenha funcionado, para remover a chave fazendo uso dos instrumentos disponíveis, a parte traseira foi danificada como mostra a Figura 27, mostrando que a remoção de componentes necessita de ferramentas adequadas.

Figura 27 – Parte traseira do kit após a remoção da chave



Fonte: O Autor

Na Figura 30, é possível ver a montagem finalizada após conectar um total de 36 pinos de propósito geral e a chave avaliada, mostrando assim que esta solução é capaz de reaproveitar pinos de entrada de chaves que apresentaram defeitos, bem como os kits que apresentaram defeito nos componentes de saída também.

4.5 Comunicação e protocolos

Na Figura 8 presente na seção 4.1, foi mostrado um diagrama de como a presente solução pode ser escalada para funcionar com múltiplos *kits* de desenvolvimento, sendo que as ligações do microcontrolador e do kit de desenvolvimento são realizadas por meio de conexões *USB* comuns.

Entre o dispositivo microcontrolador e o kit de desenvolvimento, as ligações são realizadas utilizando *jumpers* pino a pino, passando por uma interface elétrica com função de regular a tensão e proteger contra sobrecarga de corrente, como avaliado na seção 4.4. O mapeamento de todos os pinos é realizado na base de dados através da tabela *'device_pins'*.

Existem restrições que devem ser levadas em consideração ao fazer a ligação de múltiplos kits de desenvolvimento, isto porque apesar da interface entre os dispositivos *Raspberry PI* e dos demais serem realizadas por meio de conexões *USB* - que podem ser aumentadas com o auxílio de dispositivos *HUBs*, é importante atestar quanto a necessidade de fornecimento de energia do microcontrolador.

4.5.1 Possíveis contornos às restrições

O kit de desenvolvimento, diferente do microcontrolador, é alimentado normalmente na rede elétrica através de um relê controlado pelo microcontrolador. Já o fornecimento de energia do microcontrolador, no presente trabalho, foi realizado através do próprio *Raspberry PI*, isto devido ao baixo consumo energético associado a este dispositivo.

Com isto em mente, há duas soluções para aumentar a quantidade de kits controlados. A primeira é simplesmente ligar os microcontroladores na rede elétrica. Contudo, embora este método seja bastante simples, existem desvantagens. Caso o microcontrolador deixe de responder aos comandos do *Raspberry PI*, seja devido a um *overflow* de memória ocasionado por algum trecho incoerente de código ou devido a uma brusca variação na tensão fornecida, será necessário presencialmente reiniciar os dispositivos. Este problema podia ser contornado anteriormente apenas fazendo a reinicialização do *Raspberry PI*, que iria interromper o fornecimento de energia das *USB*.

De modo a contornar este problema, sugere-se o uso de um novo relê para fazer o controle dos microcontroladores. Como estes não tem um elevado consumo energético, é possível ligar todos ao mesmo tempo através de uma régua elétrica controlada por relê. Um exemplo de construção de uma régua elétrica controlada por relê está na Figura 28. Esta régua foi utilizada para controlar a ligação do kit de desenvolvimento e foi construída em cima de uma extensão de tomadas antiga da marca *SMS*.

Figura 28 – Interior de uma régua elétrica controlada por relê

Fonte: O Autor

Este novo relê deverá agora ser controlado pelo *Raspberry PI* diretamente através de sua interface *GPIO*. Para isto será necessário uma nova aplicação simples para operar junto do serviço *systemd* de modo a ligar a régua quando o serviço estiver em execução, tornando-se possível reiniciar os microcontroladores de forma remota.

Supondo que todas as necessidades elétricas sejam garantidas, é atestado pelo desenvolvedor que, a aplicação *VirtualHere* é capaz de compartilhar até 122 dispositivos *USB* distribuídas em uma profundidade de até 6 *hubs* (VIRTUALHERE, 2021).

Apesar da alta capacidade de compartilhamento da solução *VirtualHere*, um microcomputador, como o *Raspberry PI*, não terá poder de processamento suficiente para atingir este limite superior. Contudo, através de testes realizados no início do trabalho, onde utilizava-se trocas de mensagens a taxas altíssimas e com mais de um dispositivo atuando, foi possível chegar em um total de 6 dispositivos remotos simultâneos sem gerar gargalo significativo na aplicação. Talvez com a inclusão de um tempo de resposta mínimo de *2ms*, seja possível atingir mais dispositivos, entretanto, como não foram realizados testes práticos, isso não é garantido.

Há muito o que otimizar na solução atual, contudo, outra estratégia seria fazer uso de uma tecnologia capaz de realizar processamento paralelo eficiente, de modo a fazer uso de todos os núcleos de processamento do dispositivo. Desta forma, fazer uso da linguagem *C++* com a biblioteca *Boost::asio* para fazer o serviço de manipulação de entrada e saída teriam ganhos de desempenho muito mais expressivos.

4.5.2 Protocolos da aplicação

O protocolo na camada de aplicação do serviço *Web* é o *HTTP* padrão da literatura, contudo o protocolo da camada de aplicação do serviço de manipulação de entradas e saídas foi desenvolvido especialmente para este propósito. Desta forma, esta seção irá detalhar seu funcionamento. Será tratado também, as similaridades deste novo protocolo, com sua versão que opera sobre a comunicação serial com os microcontroladores.

4.5.2.1 Protocolo de comunicação pela rede

A ideia de criar um novo protocolo da camada de aplicação para o serviço de manipulação de entradas e saídas era otimizar o máximo possível o tráfego de rede e o número de pacotes enviados. Com pacotes suficientemente pequenos, é possível evitar a quebra do pacote em vários com tamanho máximo definido pelo *MTU* (*Maximum Transmission Unit*) de forma a reduzir o tráfego e otimizar a velocidade.

Devido ao sistema de encapsulamento dos protocolos, os dados enviados sobre o protocolo *TCP/IP* possuem um total de 40 *bytes* de cabeçalho, sendo vinte do protocolo *IP* e os outros vinte do *TCP/IP*, com isso temos um tamanho máximo de 1460 *bytes* de mensagem, que poderá ser tratada de forma mais sucinta enviando a mensagem de forma estruturada a nível de *bytes*, ou menos como é o caso do *HTTP* (*Hypertext transfer protocol*) onde a estrutura da mensagem é definida por texto.

Deve ser lembrado que, embora o *VirtualHere* exija conexões com baixíssima

latência, o sistema foi desenvolvido para suportar múltiplos métodos de conexão, então para torna-lo mais compatível com conexões onde o tráfego é limitado foi escolhido também o uso de um protocolo estruturado a nível de *bytes* de modo a poupar tráfego. Esta mesma estrutura será reaproveitada para ser utilizada na comunicação serial com o microcontrolador, assim fazer envio de dados de forma sucinta tornará a comunicação mais rápida em detrimento da velocidade reduzida do protocolo *UART*.

Na Tabela 6 está descrito como é a estrutura padrão de uma mensagem que utiliza este protocolo. Percebe-se que já nos primeiros quatro *bytes* são informados o tamanho total da mensagem a ser lida. Esta informação é importante porque dá um parâmetro para o sistema avaliar até qual posição dos dados escritos no *buffer* recebido são de uma mensagem, ou de outra. No protocolo *HTTP* não se utiliza o tamanho da mensagem, mas sim uma sequência de quatro caracteres ao término, contudo, é possível que um usuário mal intencionado envie uma grande quantidade de *bytes*, sem nunca colocar a sequência de finalização da mensagem de modo a tentar indisponibilizar o serviço.

Tabela 6 – Estrutura padrão do protocolo

size	opcode	message
32 bits	8 bits	msg_size bits

Fonte: O Autor

Junto ao tamanho da mensagem, o cabeçalho recebe também um *byte* de *opcode*. É este *byte* que irá direcionar o motivo da mensagem, e com isso é possível mapear os dados esperados de forma específica para cada tipo de mensagem.

Dentre os dados que podem ser adicionados na mensagem estão: números inteiros de 1 a 8 *bytes*, números de ponto flutuantes (*32-bit / IEEE 754*) e *strings*.

A *strings* (conjunto de caracteres) são enviados de uma forma diferente neste protocolo, isto porque é importante compreender a quantidade de caracteres do conjunto. Geralmente em memória *strings* são tratadas de forma a sempre finalizarem com *byte* zero, contudo, para deixar mais genérico o uso de *strings* no protocolo, de modo a conseguir enviar qualquer caractere (inclusive o *byte* zero), sempre que uma *string* for escrita, primeiramente escreve-se o tamanho dela em formato de números inteiros positivos formados por *16 bits* e em seguida a sequência.

Na Tabela 7 foi descrito um exemplo da estrutura de mensagem inicial responsável por dar início a conexão (*opcode* igual a um). Nesta mensagem são enviados dois números de um único *byte*, e um conjunto de caracteres.

Tabela 7 – Estrutura da mensagem de início da conexão

size	opcode	method_type_id	method_id	token_str_size	token_str
32 bits	8 bits	8 bits	8 bits	16 bits	token_str_size bits

Fonte: O Autor

Devido a presença de um conjunto de caracteres, o tamanho da mensagem irá variar com a *string* enviada, contudo, visto que o dado é um *token* de autenticação, sabemos de antemão que este é descrito por 16 caracteres, então esta mensagem terá um tamanho de mensagem esperado fixo em 25 *bytes*, sendo 20 efetivamente do conteúdo e os outros 5 do cabeçalho.

No trecho mostrado em Código 4 é possível avaliar a facilidade de se escrever esta mensagem utilizando a linguagem *C++* na aplicação *WiRED Panda*, e o consumo na linguagem *Python* pode ser visto no trecho do Código 5.

```
1 NetworkOutgoingMessage msg(1);
2 msg.addByte<uint8_t>(deviceTypeId);
3 msg.addByte<uint8_t>(methodId);
4 msg.addString(QString::fromStdString(token));
5 msg.addSize();
```

Código 4 – *Trecho* do código responsável por montar a mensagem

É percebido que durante a montagem da mensagem, depois de escrever todos os dados chama-se o método *addSize* que irá adicionar a quantidade de *bytes* que foram escritos até então. Esta adição será feita no início da mensagem.

```
1 device_type_id = imsg.pop_unsigned_byte()
2 method_id = imsg.pop_unsigned_byte()
3 token = imsg.pop_string()
```

Código 5 – *Trecho* do código responsável por consumir a mensagem

Como as mensagens recebidas pelo protocolo *TCP/IP* estarão sempre completas e íntegras - devido aos mecanismos de correção e ordenação do protocolo, torna-se trivial fazer a leitura destas mensagens, sendo necessário após o recebimento apenas direcionar com base no *opcode* e ir desempilhando os dados na mesma ordem em que foram empilhados (por se tratar de uma fila). Será visto na próxima subseção 4.5.2.2 que receber as mensagens nem sempre é trivial.

4.5.2.2 Protocolo de comunicação serial

De modo a aproveitar a mesma estrutura anterior, os pacotes são montados de forma semelhante, contudo desta vez estes são enviados por um meio suscetível a erros, mas sem implementações na camada de rede para corrigi-los.

Diferente do que acontecem com pacotes do tipo *UDP*, não é possível que pacotes cheguem em ordem errada, mas estes podem chegar corrompidos, e assim é importante que o sistema tenha mecanismos para descartar os pacotes errados assim como mecanismos para fazer o realinhamento da leitura.

O padrão da comunicação *UART* (*Universal asynchronous receiver/transmitter*) utilizado foi o *8E2*, onde cada símbolo é formado por 11 *bits*, sendo os 8 primeiros de dados seguidos pelo *bit* de paridade e dois *bits* de parada. Como o padrão utilizado foi o padrão par (*Even*), então se o *bit* for zero, significa que no dado havia um número par de *bits* com valor um, caso fosse um número ímpar de *bits* com valor um então o *bit* de paridade seria um. No padrão ímpar, a lógica é invertida, ou seja, *bit* de paridade zero, significa número *ímpar* de *bits* um.

É visto que esta verificação de paridade, embora ajude, não é capaz de determinar que a mensagem é íntegra, isto porque faz verificação símbolo a símbolo. Além disso, por utilizar somente um *bit* de paridade, em caso de erro em dois *bits* de um mesmo símbolo, estes serão avaliados corretamente pela paridade, desta forma, este método não descarta a necessidade de uma verificação de paridade da mensagem como um todo.

Além de fazer uso deste padrão, foi necessário garantir a possibilidade de realinhar a leitura, isto porque caso o primeiro símbolo (neste caso *byte*) seja avaliado como errado pelo sistema de paridade e ele inicie sua leitura a partir do segundo *byte*, o tamanho da mensagem será deslocado. Ou seja, supondo que o tamanho da mensagem tenha 1 *byte* somente, este *byte* será descartado e o segundo *byte* que seria o *opcode* da mensagem, irá ser lido como o tamanho, logo, haverá um problema que poderá repercutir daí em diante.

De modo a realinhar a leitura, foi adicionado um cabeçalho de 10 *bytes* no início de todas as mensagens. Este cabeçalho é estático e é importante para que a leitura seja sincronizada.

É possível reduzir drasticamente a taxa de erro apenas selecionando um *baudrate* que seja melhor compatível com a frequência do *clock* do microcontrolador. A placa genérica usada possui uma frequência de 16 *MHz*. No *datasheet* do microcontrolador (ATMEL CORPORATION, 2014) é possível encontrar a relação da frequência utilizada e a taxa de erro dos dados trocados. Na Tabela 8 está presente esta relação.

Avaliando a tabela, mais especificamente na frequência utilizada, é possível avaliar que a *baudrate* com menor taxa de erro são as de *250K*, *0.5M* e *1M*. Neste caso, a taxa escolhida para o sistema foi a de *250K*, que pode ser traduzida para 250 mil símbolos por segundo.

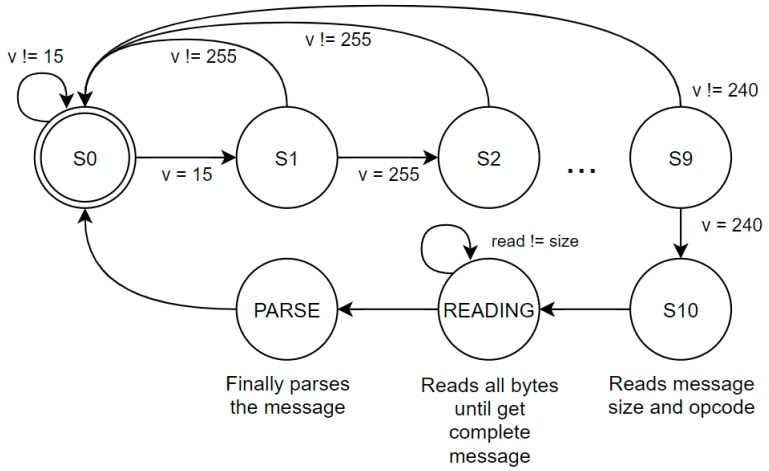
Tabela 8 – Exemplos de configurações *UBRRn* para as frequências de osciladores mais comuns

Baud Rate [bps]	f_{osc} = 16.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4K	68	0.6%	138	-0.1%
19.2K	51	0.2%	103	0.2%
28.8K	34	-0.8%	68	0.6%
38.4K	25	0.2%	51	0.2%
57.6K	16	2.1%	34	-0.8%
76.8K	12	0.2%	25	0.2%
115.2K	8	-3.5%	16	2.1%
230.4K	3	8.5%	8	-3.5%
250K	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max.(1)	1Mbps		2Mbps	

Fonte: ATMEL Corporation (2014)

A máquina de estados que descreve a leitura dos *bytes* esta descrita na Figura 29. Perceba que a leitura do conteúdo da mensagem é somente realizada após fazer a leitura dos *bytes* do cabeçalho, e caso algum não esteja de acordo com a ordem previamente especificada, então o sistema irá voltar a leitura para o estado inicial. Com isto, o sistema será capaz, em caso de erro, sincronizar novamente o envio de mensagens.

Figura 29 – Máquina de estados de leitura de mensagem



Fonte: Criado pelo autor através da ferramenta *DrawIO*.

Nos testes realizados não houve a necessidade de realizar uma verificação da

integridade da mensagem (devido a baixa taxa de ocorrência de erros), contudo isto seria algo importante a ser implementado para buscar melhor garantir que a instrução enviada esteja íntegra.

4.6 Infraestrutura necessária

Devido as restrições aplicadas a conectividade da rede da universidade, tornou-se mais burocrático o processo de obtenção de acesso a *VPN* (*Virtual Private Network*), e devido a isso, poucos seriam os alunos que poderiam fazer uso do sistema.

Para solucionar este problema foi feito uso de uma infraestrutura *cloud* para direcionar os pacotes de acesso externo de modo controlado para dentro da rede. Este direcionamento foi realizado fazendo uso do *software VPN WireGuard*, um projeto de código aberto licenciado por um conjunto de licenças compatíveis com a *GPL*.

A provedora do serviço foi a *Google Cloud*. O serviço foi contratado por meio de sua plataforma *on-line*, e a hospedagem escolhida está situada em um *datacenter* na cidade de São Paulo.

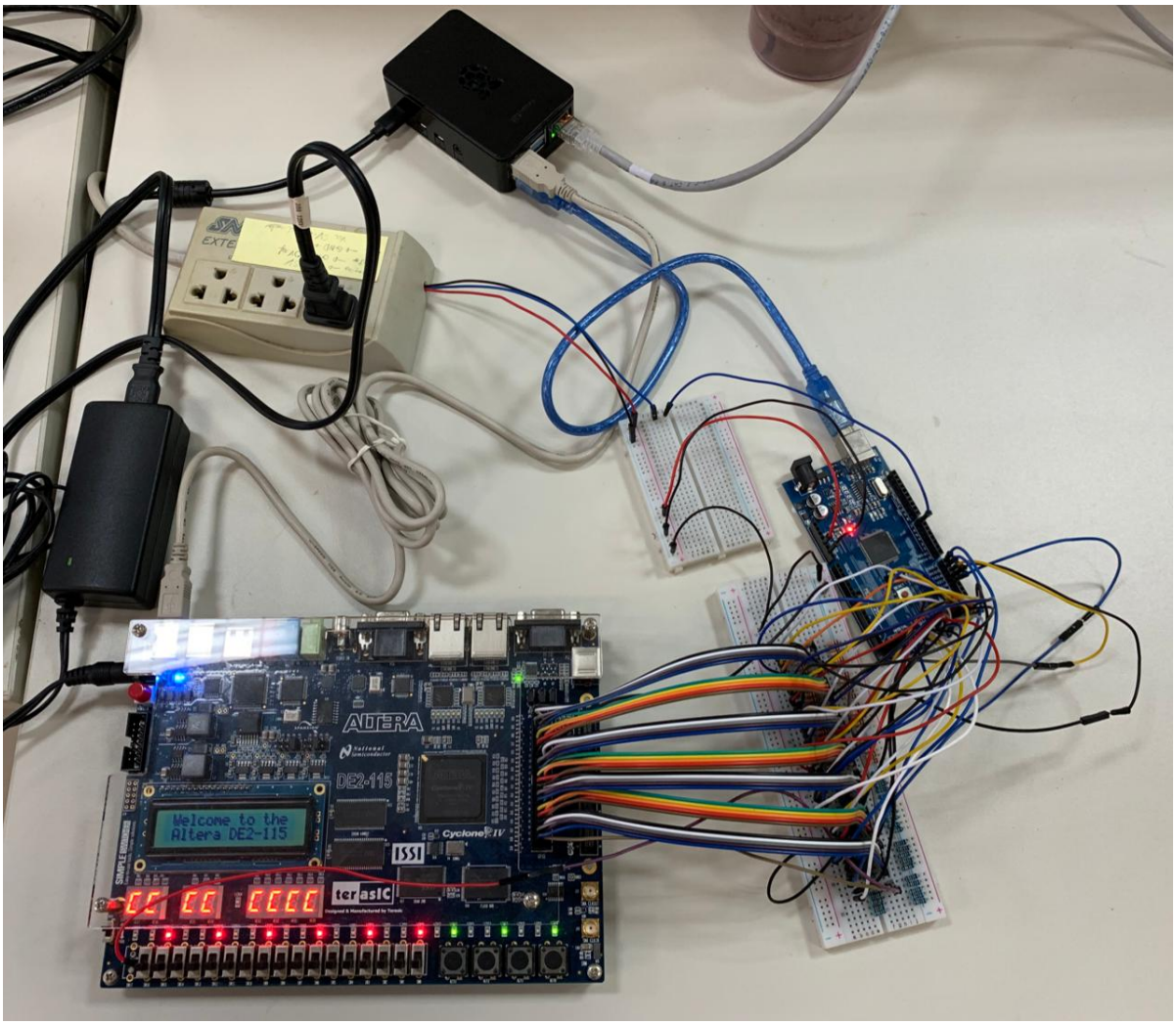
O *software WireGuard* tem sido reconhecido por sua fácil implantação, por estar presente em diversas plataformas e devido seu protocolo possuir um desempenho tão bom quanto os outros protocolos da literatura (*OpenVPN* e *IPSec*). Enquanto os demais protocolos possuem abordagens mais complexas e estão em uso há muito tempo, o *WireGuard* além de ser recente, provou-se escalável e extremamente seguro mesmo tendo seu foco direcionado a simplicidade e usabilidade (DONENFELD, 2017).

Foi possível fazer os redirecionamentos dos pacotes específicos para o uso dos serviços deste trabalho utilizando o redirecionador de pacotes do *Linux iptables*. Fazendo uso de uma arquitetura simples com apenas dois nós, o servidor *cloud* foi configurado como provedor de acesso para o serviço instalado dentro da universidade.

5 Resultados e Discussões

É mostrada na Figura 30 a foto da instalação do sistema na Universidade Federal de São Paulo. Com o sistema em funcionamento, foram realizados alguns testes para classificar a latência fazendo uso do sistema de redirecionamento com serviço *cloud*, e também através da *VPN* da Universidade. Além dos testes para avaliar o tempo de responsividade da solução, durante o período de uma semana, alunos fizeram uso da solução e responderam ao questionário comentado na seção 3.2.

Figura 30 – Montagem do laboratório



Fonte: O Autor

5.1 Testes de responsividade da solução

Utilizando o próprio serviço, foi avaliada a latência de diferentes cidades, com diferentes formas de conexão, utilizando ou não o acesso externo (que faz uso do serviço de redirecionamento de pacotes). Os resultados foram inseridos na Tabela 9.

Tabela 9 – Montagem do laboratório

Localização	Latência Média (ms)	
	Externo	Interno
Interior de São Paulo	15	13
São José dos Campos	14	11
São Paulo	8	8

Fonte: O Autor

O acesso interno é feito utilizando o serviço *VPN* da própria Universidade, contudo, este também está presente na cidade de São Paulo, então devido a isto, não é possível avaliar grandes diferenças na latência ao fazer acesso através do provedor de acesso *cloud* que está situado na mesma cidade.

Alguns pontos a serem considerados:

- A medição avaliada como *Interior de São Paulo* foi realizada com dados da cidade de *Campinas* e *São João da Boa Vista*.
- A latência foi medida através da própria ferramenta *WiRED Panda* calculando a diferença do tempo do envio de uma mensagem e sua respectiva resposta.
- Todas as medições foram realizadas utilizando provedores de *internet* com tecnologia *FTTH* (*Fiber to the home*), outras formas de conexão irão impactar diretamente na latência.

Estes resultados são importantes devido a existência de uma latência máxima recomendada para uso da solução *VirtualHere*, esta, segundo o desenvolvedor, é de 35 *ms*. Com os resultados obtidos, é possível garantir o funcionamento deste método de conexão com uma boa margem, isto para conexões *FTTH*.

Foram realizados testes de usabilidade utilizando conexão móvel de tecnologia *4G*, foi possível obter uma latência média de 50 *ms*. Mesmo apresentando uma latência superior a recomendada (de 35 *ms*), foi possível fazer uso da solução normalmente. Contudo foi assim avaliado a necessidade de um maior tempo para realizar a programação no dispositivo.

Testes com conexão móvel de tecnologia *3G* foram realizados também, contudo devido a alta instabilidade e grandes variações na latência, não foi possível fazer uso da

solução *VirtualHere*, abrindo precedentes para trabalhos futuros que envolvam a integração com outro método de conexão para atender aos alunos com conexões de alta instabilidade.

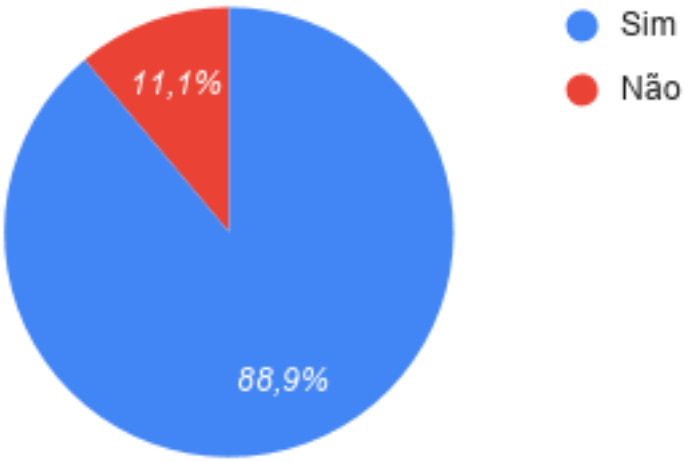
Apenas a critérios comparativos, a velocidade com que um pacote é enviado e retornado na pior medição (15 *ms*), é inferior a velocidade de atualização da imagem de um monitor convencional (que opera a uma taxa de 60 *Hz*) algo que irá agregar uma altíssima responsividade ao sistema.

Embora a latência medida seja extremamente baixa, o tempo necessário para atualizar uma saída, dado a uma mudança na entrada, pode contudo, corresponder a um tempo ligeiramente maior, isto devido ao tempo necessário para o microcontrolador fazer a alteração do valor de entrada, e notificar a mudança na saída para o *Raspberry PI*.

5.2 Resultado do questionário

Em razões de curto prazo e inviabilidade de datas para a instalação do sistema na Universidade, o questionário esteve presente para preenchimento durante apenas uma semana, o que levou a poucas avaliações. Durante os sete dias de pesquisa foram obtidos um total de 9 avaliações. Dos 9 alunos que responderam ao questionário, apenas um não foi capaz de utilizar o sistema como mostra a Figura 31.

Figura 31 – Alunos que conseguiram fazer uso do sistema



Fonte: O Autor

O comentário do aluno que não foi capaz de utilizar a solução indicou que o problema se deu apenas no uso do dispositivo de forma remota, e que o dispositivo não aparecia na lista do *software Quartus*. É possível que isto tenha ocorrido em detrimento de uma conexão instável e com latências superiores a 50 *ms*, entretanto, como a resposta deste aluno quanto a tecnologia utilizada foi *Wi-fi, com internet FTTH (fibra ótica)*, pode-se talvez, descartar esta possibilidade.

Foi relatado por um docente um erro na instalação do *software Quartus*, onde apenas o *driver* do dispositivo *USB-Blaster* não era instalado, e isto ocasionou exatamente o mesmo erro ao fazer uso do dispositivo, mesmo conectado de forma física. Com isto em mente, acredita-se que o problema do aluno tenha sido o mesmo, visto que além de apresentar uma boa conexão, este encontra-se no primeiro laboratório da trajetória acadêmica, mostrando que provavelmente tenha feito a instalação do *Quartus* recentemente.

Devido a incapacidade de utilizar o sistema por motivos técnicos, aqueles que não conseguiram fazer o uso não foram questionados quanto as questões de usabilidade, sendo assim, das 9 avaliações, haviam apenas 8 respostas referente as perguntas do *SUS*.

A solução foi pontuada pela pontuação *SUS* com nota $94,69$, algo que se encaixaria no adjetivo *Melhor imaginável* (BANGOR; KORTUM; MILLER, 2009). Segundo (NUNNALLY, 1978) é necessário ao menos a participação de cinco pessoas para assegurar uma estimativa estável da pontuação. Entretanto, quanto maior o número de avaliações, maior será a confiança atrelada a esta pontuação. As respostas obtidas estão apresentadas na Tabela 10.

Tabela 10 – Respostas da perguntas do *SUS*

	Perguntas	Respostas				
		1	2	3	4	5
1.	Acredito que gostaria de utilizar este sistema frequentemente.	1	0	1	1	5
2.	Achei o sistema desnecessariamente complexo.	7	1	0	0	0
3.	Achei o sistema fácil de usar.	0	0	1	1	6
4.	Acredito que precisaria do apoio de um suporte técnico para fazer uso do sistema.	7	0	1	0	0
5.	Achei que as funções presentes neste sistema foram bem integradas.	0	0	0	0	8
6.	Achei que houve muitas inconsistências neste sistema.	8	0	0	0	0
7.	Imagino que a maioria das pessoas aprenderiam a usar esse sistema rapidamente.	0	0	0	2	6
8.	Achei o sistema muito pesado para utilizá-lo.	7	1	0	0	0
9.	Me senti bastante confiante usando esse sistema.	0	0	0	1	7
10.	Houve a necessidade de aprender uma série de coisas antes que eu pudesse continuar a utilizar esse sistema.	8	0	0	0	0

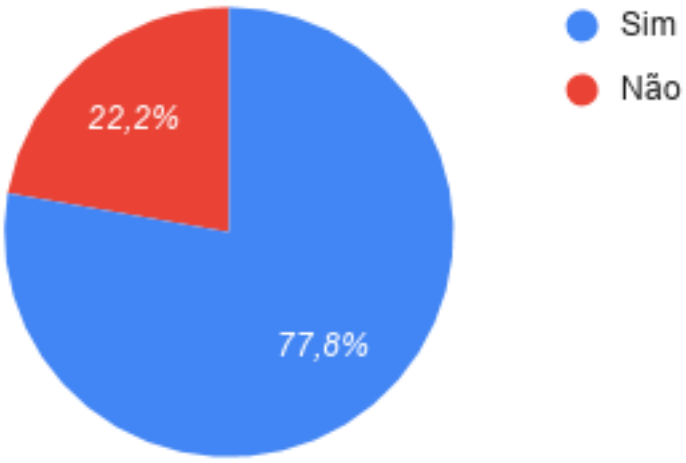
Fonte: O Autor

Além das perguntas referentes ao questionário do *SUS*, algumas outras perguntas foram adicionadas como descritas na seção 3.2.

Foi realizado também um levantamento quanto ao uso prévio do simulador *WiRED Panda*, isto devido ao fato de que alunos que já fizeram uso do simulador estão acostumados a utilizá-lo, o que irá impactar no questionário de usabilidade.

Dos 9 alunos que participaram da pesquisa, 7 deles tiveram contato com o simulador (vide Figura 32). Esta alta quantidade de alunos se dá devido ao uso deste simulador por alguns docentes que lecionam a disciplina de *Circuitos Digitais* na Universidade em questão.

Figura 32 – Alunos que já tinham tido contato com o simulador



Fonte: O Autor

Ao avaliarmos a pontuação *SUS* apenas daqueles que tinham tido contato com o simulador, obtemos uma pontuação de $93,75$. Embora tenham sido realizadas poucas avaliações, esta análise poderia sugerir que a própria usabilidade do simulador tenha tido impacto na pesquisa, isto porque os alunos que utilizavam previamente a ferramenta, por já estarem acostumados a fazer uso desta, devem focar a avaliação na extensão do laboratório remoto ao invés de re-avaliar toda a ferramenta.

Como a pontuação obtida do sub-conjunto de respostas foi inferior, isto pode demonstrar que a usabilidade do simulador têm sido maior que a usabilidade da extensão do laboratório remoto. Realmente, dentre as sugestões inseridas, há duas sugestões para melhorar a usabilidade do sistema, sendo uma delas a correção de um *bug* que fazia a janela de autenticação fechar após iniciar a conexão ou ser direcionado a fila - fazendo necessário ao usuário abrir novamente o menu de configurações. A outra sugestão foi referente a parte de realização da pinagem, que no caso, não aceita o clique duplo como forma de seleção rápida do pino da lista, fazendo com que o usuário tenha que selecionar

o pino e em seguida pressionar o botão *Ok*.

Outras sugestões apresentadas pelos alunos foram:

- Aumentar a quantidade de pinos para utilização;
- Adicionar o componente do *display LCD* no simulador;

É possível fazer o aumento da quantidade de pinos ao substituir um maior número de chaves físicas do kit de desenvolvimento *DE2-115*, ou fazendo uso de um outro kit de desenvolvimento com uma maior quantidade de pinos disponíveis para propósito geral. Entretanto deve-se avaliar que o microcontrolador utilizado no momento é um *ATmega 2560*, e este embora apresente uma quantidade grande de pinos, também é um fator limitante (máximo de 54 pinos digitais e 16 pinos analógicos).

Quanto ao *display LCD*, é possível adiciona-lo ao simulador visto que este opera de forma *assíncrona*, sendo compatível com esta solução. Entretanto, será necessário configurar o dispositivo para que o pulso do *Enable* dure um tempo maior, e que tenha duração suficiente para ser reconhecido pelo microcontrolador e repassado para o usuário. A implementação porém, não é trivial, isto devido aos inúmeros comandos a serem implementados e funcionalidades que um componente físico possui suporte.

Foram duas as críticas ao sistema:

- Ter de fazer uso do *Quartus*;
- Resolver falhas do simulador já conhecidas, como por exemplo, aplicação fechando sozinho sem salvar o conteúdo no momento de realizar ligação de fios;

A ferramenta *Quartus* é bastante pesada e apresenta uma quantidade considerável de *bugs*, e infelizmente não existe uma ferramenta alternativa para a programação de *FPGAs* da *Altera*. Quanto às falhas do simulador, estas podem talvez até estarem resolvidas, isto porque esta versão foi desenvolvida baseada em uma versão já ultrapassada do simulador.

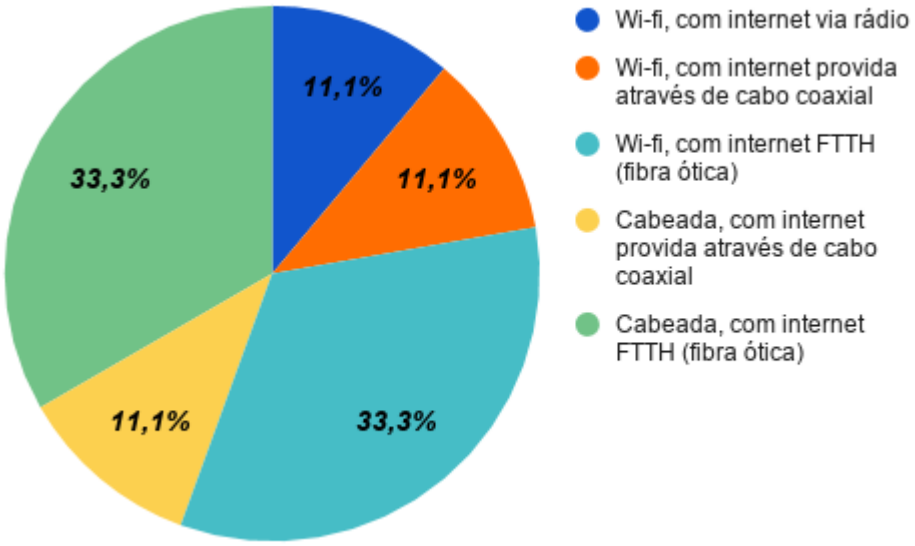
Foram elogiados os seguintes detalhes da implementação:

- A implementação da fila;
- A integração do sistema remoto com o simulador;

Outro levantamento foi realizado para avaliar a tecnologia de conexão utilizada pelos alunos, afinal, a tecnologia da conexão está diretamente relacionada com a latência e esta que contribui para uma melhor responsividade, que por sua vez é uma qualidade avaliada na usabilidade de um sistema.

Os resultados obtidos estão presentes na Figura 33. É possível perceber que em geral a tecnologia de conexão utilizada por alunos e ex-alunos tem sido tecnologias compatíveis com o sistema. Entre as tecnologias utilizadas, a pior avaliada é a tecnologia a rádio, isto porque esta tende a possuir uma latência consideravelmente maior e que pode apresentar instabilidades mediante a condições climáticas.

Figura 33 – Tecnologia de conexão a *internet* utilizada



Fonte: O Autor

Duas outras perguntas na escala *Likert* foram realizadas, e os resultados obtidos estão disponíveis na Tabela 11.

Tabela 11 – Respostas da perguntas adicionais

	Perguntas	Respostas				
		1	2	3	4	5
1.	Acredito que o momento de disponibilização desta ferramenta é oportuno.	0	0	2	0	7
2.	Acredito que esta ferramenta torna as possibilidades de desenvolvimento de circuitos digitais menos limitadas, contribuindo positivamente para o desenvolvimento das atividades.	0	0	1	0	8

Fonte: O Autor

A primeira delas avalia a compatibilidade do projeto com o contexto sendo vivenciado. O alto índice de respostas positivas pode estar relacionado com o fato de o ensino

estar sendo realizado apenas de forma remota, devido ao contexto atual na ocasião da escrita desta monografia..

O outro ponto avaliado refere-se a perspectiva dos alunos sobre as possibilidades que esta ferramenta pode trazer para a execução das atividades. Avaliado na escala *Likert*, apenas um aluno foi indiferente, sendo que os demais pontuaram com o topo da escala '*Concordo totalmente*'. A ideia geral desta pergunta foi avaliar a compreensão dos alunos quanto a nova proposta, de forma que os alunos que avaliaram positivamente perceberam que tornando os dispositivos de entrada e saída desacoplados do kit de desenvolvimento, estes podem fazer uso de componentes antes não disponíveis para uso e não somente isto, mas também que podem distribuir os pinos entre diferentes componentes conforme a necessidade, não sendo limitado a utilizar somente os recursos presente no kit.

A ideia geral desta pergunta foi avaliar a compreensão dos alunos quanto a nova proposta, podendo concluir que os alunos perceberam que tornando os dispositivos de entrada e saída desacoplados do kit de desenvolvimento, estes podem fazer uso de componentes antes não disponíveis para uso. Além disso, os alunos perceberam que podem distribuir os pinos entre diferentes componentes conforme a necessidade, não sendo limitado a utilizar somente os recursos presente no kit.

6 Considerações finais

No presente trabalho foi possível desenvolver um método alternativo para a utilização do kit de desenvolvimento *DE2-115* de forma remota. A solução foi desenvolvida de forma genérica e acoplada ao simulador de circuitos digitais *WiRED Panda*, sendo não somente escalável, mas podendo ser utilizada com outros kits de desenvolvimento com ou sem o uso de componentes *FPGA*.

Com um desenvolvimento focado em oferecer uma melhor usabilidade, este trabalho também teve como foco manter fiel o uso do *kit de desenvolvimento* como se estivesse no laboratório, contudo oferecendo uma forma alternativa de manipular o *kit* e conferir os resultados.

Fazendo uso do *software* de prateleira *VirtualHere* de forma integrada a solução, foi possível aproximar o uso do laboratório de forma remota ao uso presencial, não limitando o aluno a funcionalidade única de realizar a re-programação.

Além de desenvolvida, a nova solução foi instalada na Universidade e foi disponibilizada para os alunos e ex-alunos realizarem testes e validarem o funcionamento. Uma pesquisa foi realizada com estes alunos de modo a avaliar questões de usabilidade e contextuais.

Embora o número de participações na pesquisa não tenha sido expressivo, foi possível obter resultados bastante positivos quanto a execução deste trabalho. Além disso, foi possível validar o funcionamento desta solução, que diferentemente da solução apresentada anteriormente, pode ser classificada não somente como um laboratório remoto, mas também como um laboratório híbrido que trás as funcionalidades de um simulador e as integra com um laboratório remoto (RODRIGUEZ-GIL et al., 2016).

As principais vantagens do resultado final ter sido de um laboratório híbrido se dá justamente na possibilidade do aluno segregar parte da lógica no simulador, e a outra parte no componente controlado remotamente como realizado no exemplo da Figura 20.

De modo a garantir a continuidade do trabalho, todos os componentes necessários foram doados a Universidade e o acesso aos alunos será mantido, porém somente de forma interna (utilizando a *VPN* da Universidade).

6.1 Problemas encontrados

Não foram poucos os empecilhos encontrados no caminho deste desenvolvimento, visto que foi necessário a disponibilização de uma infraestrutura *cloud* somente para

garantir acesso externo aos alunos durante o período de avaliação do sistema. O uso desta infraestrutura foi de suma importância também durante o desenvolvimento, devido a necessidade de realização de testes a distância e ao fato do serviço precisar ser disponibilizado mesmo atrás de um *NAT* devido a restrições do provedor. O serviço *VPN* da Universidade se mostrou muito instável durante o desenvolvimento do trabalho, e fazendo uso do provedor *Conexão Telecom* em São João da Boa Vista, a latência utilizando o serviço tinha média de 110 *ms*, impossibilitando seu uso.

Não foi possível compreender ao certo o motivo do alto tempo de resposta ao utilizar o serviço da *VPN* usando este provedor, visto que fazendo o uso da *VPN* disponibilizada pelo serviço *cloud* este problema não ocorria, talvez seja simplesmente um direcionamento por uma rota com gargalo ou uma política de limitação de tráfego *UDP* da porta 4501 utilizada pelo protocolo *IPSec*.

6.2 Próximos passos

Como próximos passos, deve-se instalar novos kits de desenvolvimento a esta solução de modo a melhor atender a quantidade de alunos, e fazer uso desta aplicação em outros momentos da formação do aluno, como por exemplo, no Laboratório de Sistemas Embarcados que faz uso de kits de desenvolvimento *PIC* e Micro-controladores *Arduino*.

Embora muitos tenham sido os testes, como estes fizeram uso de somente um kit de desenvolvimento, não foi possível garantir o funcionamento da solução para o controle de múltiplos kits, então é sugerido realizar novos testes ao fazer uso de mais de um kit de desenvolvimento de forma simultânea.

O desenvolvimento do simulador *WiRED Panda* também deve ser encorajado, visto que este irá proporcionar mais componentes e casos de uso para o laboratório híbrido. Um painel administrador poderá também ser desenvolvido para facilitar a manipulação das configurações de dispositivos e pinos, assim tornando mais fácil a adoção do sistema em outras escolas.

Referências

ALTERA CORPORATION. *Cyclone IV, Device Handbook*. [S.l.]: March, 2016. Citado 2 vezes nas páginas 19 e 55.

ÁLVARES, A. J.; FERREIRA, J. C. E. Metodologia para implantação de laboratórios remotos via internet na área de automação da manufatura. In: *2o Congresso Brasileiro de Engenharia de Fabricação (COBEF), Uberlândia, MG*. [S.l.: s.n.], 2003. v. 18. Citado na página 11.

APPLE INC. *Placa Afterburner*. [S.l.], acessado em 22 de fevereiro de 2021. Disponível em: <https://www.apple.com/br/shop/product/MW682AM/A/placa-afterburner-da-apple>. Citado na página 19.

ATMEL CORPORATION. *8-bit Microcontroller with 256K Bytes In-System Programmable Flash*. [S.l.], 2014. Citado 3 vezes nas páginas 55, 63 e 64.

BANGOR, A.; KORTUM, P.; MILLER, J. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, Citeseer, v. 4, n. 3, p. 114–123, 2009. Citado 2 vezes nas páginas 29 e 70.

BERALDO, A. L. da S.; OLIVEIRA, T. d. O. T. de; STRINGINI, D. Laboratórios remotos de fpga com foco no ensino: Uma revisão sistemática da literatura. *RENOTE*, v. 18, n. 1, 2020. Citado na página 20.

BOULDER, U. of C. *About PhET*. [S.l.], (acessado em 22 de fevereiro de 2021). <https://phet.colorado.edu/pt_BR/about>. Citado na página 15.

BRAGGIO, A. A. *FPGA: processo de configuração com desenvolvimento de protótipo em placa de circuito impresso*. Dissertação (B.S. thesis) — Universidade Tecnológica Federal do Paraná, 2014. Citado 2 vezes nas páginas 12 e 18.

BROOKE, J. Sus: a "quick and dirty'usability. *Usability evaluation in industry*, v. 189, 1996. Citado na página 28.

BRUNET, A. P.; NEW, S. Kaizen in japan: an empirical study. *International Journal of Operations & Production Management*, MCB UP Ltd, 2003. Citado na página 12.

DONENFELD, J. A. Wireguard: Next generation kernel network tunnel. In: *NDSS*. [S.l.: s.n.], 2017. Citado na página 65.

FOUNDATION, R. P. *Raspberry PI OS*. [S.l.], (acessado em 22 de fevereiro de 2021). <https://www.raspberrypi.org/documentation/raspbian/>. Citado na página 46.

FREE SOFTWARE FOUNDATION. *GNU General Public License*. 2007. Disponível em: <https://www.gnu.org/licenses/gpl.html>. Citado 2 vezes nas páginas 24 e 25.

FREE SOFTWARE FOUNDATION. *GNU Lesser General Public License*. 2007. Disponível em: <https://www.gnu.org/licenses/lgpl-3.0.html>. Citado na página 24.

HIROFUCHI, T. *README for usbip-utils*. [S.l.], 2011 (acessado em 8 de outubro de 2020). <https://www.kernel.org/doc/readme/tools-usb-usbip-README>. Citado na página 21.

HIROFUCHI, T. et al. Usb/ip-a peripheral bus extension for device sharing over ip network. In: USENIX ASSOCIATION. *Proceedings of the annual conference on USENIX Annual Technical Conference*. [S.l.], 2005. p. 42–42. Citado na página 21.

LINDSAY, E. et al. Remote laboratories in engineering education: Trends in students' perceptions. In: AUSTRALASIAN ASSOCIATION FOR ENGINEERING EDUCATION. *Proceedings of the 18th Conference of the Australasian Association for Engineering Education*. [S.l.], 2007. Citado na página 15.

LIU, C. C. Teaching digital designs by building small autonomous robotic vehicles using an fpga platform. In: *2015 ASEE Annual Conference & Exposition*. [S.l.: s.n.], 2015. Citado na página 19.

LOURENÇO, R. da S. Laboratórios remotos-um estudo para a puc-rio. 2014. Citado na página 15.

MAHNIC, V. Improving software development through combination of scrum and kanban. *Recent Advances in Computer Engineering, Communications and Information Technology, Espanha*, p. 281–288, 2014. Citado na página 21.

MARTIN, J. *Information engineering: introduction*. [S.l.]: Prentice Hall PTR, 1989. Citado na página 49.

MAYOZ, C. A. et al. Fpga remote laboratory: Experience in upna and unifesp. In: SPRINGER. *International Conference on Remote Engineering and Virtual Instrumentation*. [S.l.], 2020. p. 112–127. Citado 2 vezes nas páginas 12 e 31.

MEC. *Diretrizes Curriculares Nacionais para os cursos de graduação na área da Computação*. [S.l.], 2016. Citado na página 11.

NIELSEN, J. *Usability 101: introduction to usability. Jakob Nielsen's Alertbox*. [S.l.]: Recovered 25/03/2015 http://www. useit. com/alertbox/20030825. html, 2003. Citado na página 28.

NUNNALLY, J. C. Psychometric theory (2nd edit.). *New York*, 1978. Citado na página 70.

PINTILIE, L. N. et al. Multifunctional socket for smart grid applications using ethernet over power lines and usb over ip technologies. In: IEEE. *2018 International Conference on Applied and Theoretical Electricity (ICATE)*. [S.l.], 2018. p. 1–5. Citado na página 26.

RAJ, C. R.; TOLETY, S. B. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In: IEEE. *2012 Annual IEEE India Conference (INDICON)*. [S.l.], 2012. p. 625–629. Citado na página 22.

RASCHKE, C. A. *The digital revolution and the coming of the postmodern university*. [S.l.]: Routledge, 2003. Citado na página 11.

RODRIGUEZ-GIL, L. et al. Towards new multiplatform hybrid online laboratory models. *IEEE Transactions on Learning Technologies*, IEEE, v. 10, n. 3, p. 318–330, 2016. Citado na página 75.

Rodriguez-Gil, L. et al. Graphic technologies for virtual, remote and hybrid laboratories: Weblab-fpga hybrid lab. In: *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. [S.l.: s.n.], 2014. p. 163–166. Citado na página 16.

RS COMPONENTS LTD. *Altera FPGA EP4CE6E22C8N, Cyclone IV E 6272 Cells, 270kbit, 392 Blocks, 144-Pin EQFP*. [S.l.], acessado em 22 de fevereiro de 2021. Disponível em: <https://uk.rs-online.com/web/p/fpgas/7293757p>. Citado na página 18.

SAURO, J.; LEWIS, J. R. Correlations among prototypical usability metrics: Evidence for the construct of usability. In: *Proceedings of the SIGCHI conference on human factors in computing systems*. [S.l.: s.n.], 2009. p. 1609–1618. Citado na página 28.

SILVA, T. A. V. et al. Sistema thin client para aplicações laboratoriais de ensino. Universidade Federal de Uberlândia, 2019. Citado na página 26.

TERASIC TECHNOLOGIES INC. *DE2-115 - User Manual*. [S.l.], 2010. Citado na página 20.

VIDAL, N. F.; MENEZES, P. H. D. Laboratório real x laboratório virtual: possibilidades e limitações desses recursos no ensino de eletrodinâmica. 2015. Citado 3 vezes nas páginas 11, 15 e 16.

VIRTUALHERE. *VirtualHere*. [S.l.], 2021. Disponível em: <https://www.virtualhere.com/>. Citado 2 vezes nas páginas 25 e 60.

WIEMAN, C. E.; ADAMS, W. K.; PERKINS, K. K. Phet: Simulations that enhance learning. *Science*, Citeseer, v. 322, n. 5902, p. 682–683, 2008. Citado na página 15.

WiRED Panda Team. *WiredPanda*. 2020. Disponível em: <https://wiredpanda.org/>. Citado na página 12.

WUNDERLICH, J. Focusing on the blurry distinction between microprocessors and microcontrollers. In: *Proc. of ASEE Nat'l Conf.* [S.l.: s.n.], 1999. Citado na página 17.